MICROPOWER SERIES

INTRODUCTION TO GRAPHICS FOR THE TI-99/4A®

John P. Grillo/ J. D. Robertson/ Terry F. Zbyszynski

REQUIRED FOR THIS BOOK:
Extended Basic
Cartridge
Memory Expander
Disk Controller
and Drive

Includes 38 programs to help you explore the world of graphics.

MICROPOWER SERIES

INTRODUCTION TO GRAPHICS FOR THE TI-99/4A®

John P. Grillo *J. D. Robertson* *Terry F. Zbyszynski*

wcb
Wm. C. Brown Publishers
Dubuque, Iowa

Consulting Editor:
Edouard J. Desautels
University of Wisconsin—Madison

Micropower Series

Cover photo by Bob Coyle

Library of Congress Catalog Card Number: 84–70102

ISBN 0–697–00237–3

2–00237–01

Printed in the United States of America

To Norma and Ed

Contents

Introduction

The purpose of this book is to explore the computer's special abilities to produce graphic displays. The book is geared toward one microcomputer, the TI–99/4A Home Computer, and one computer language, BASIC. However, due to the generalized nature of the discussion, we feel that the techniques employed in the many sample programs can be adapted to other hardware and other languages.

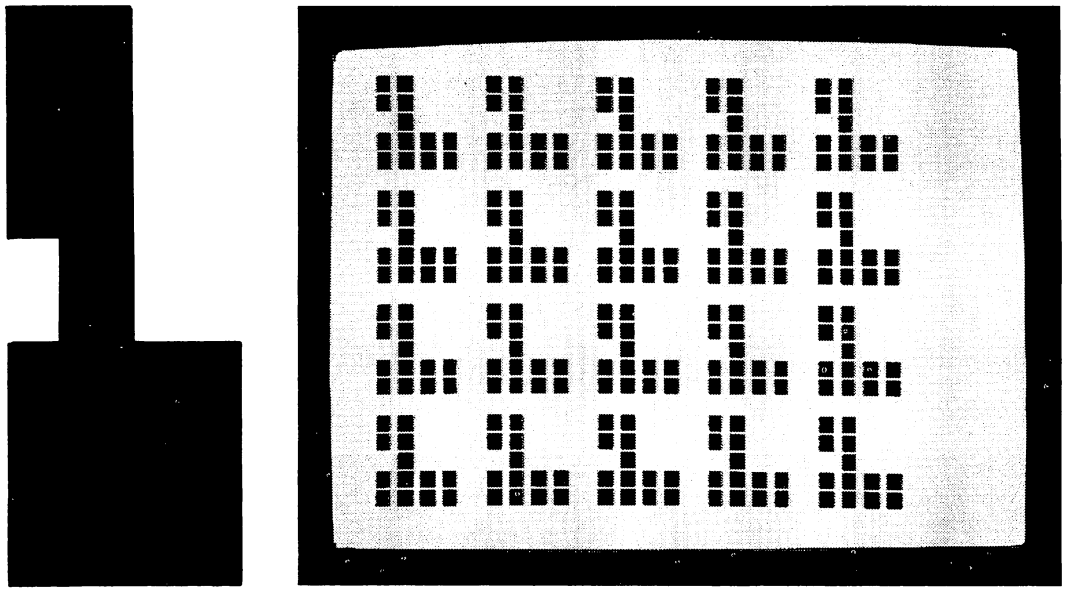
Three general methods for producing computer graphics are examined in depth. The most traditional method of creating pictures with a computer, *line printer graphics,* is treated first. With line printer graphics, the picture is considered to be a set of horizontal lines whose components are the letters, digits, and special symbols that make up BASIC's character set. This method of producing outlines, silhouettes, graphs, and other visually appealing pictures takes up the largest part of the book, as it should. Line printer graphics can be produced using any language and with any computer, with or without a printer.

Character graphics is the second method for creating graphics and relies on the TI–99/4A Home Computer's graphic symbols. Many microcomputer makers have adopted special graphic characters, and for good reason. These characters allow the video screen to display a wide variety of special effects, such as game boards and pieces, lunar landers and space ships, faces and entire bodies, even schematics and blueprints. The popularity of this technique has increased significantly since the appearance of the many microcomputer chess games on the market in the late 1970's.

The third technique for producing graphics is sprite graphics. Sprites are graphics which have color and a location anywhere on the screen. They can be set in motion and move more smoothly than the usual character. Used with MAGNIFY or MOTION they can change size or movement. The microcomputer programmer who can deal with sprite graphics on the TI–99/4A Home Computer can produce computer displays with smoother displays than can be produced with character graphics.

In this book, we have limited our discussion to low-resolution graphics only. We do not discuss the color, sound, joystick, and lightpen features of this fine machine. We hope to cover these topics in a subsequent book. Also, you will notice our frequent use of the letters TI–99/4A instead of the full name, Texas Instrument 99/4A Home Computer. This shorthand version of the name has become commonly accepted.

As you proceed through the discussion of the two graphics methods, you will discover a diversity of techniques exemplified by many complete programs, both short and long. We hope that you will try them out and modify them as you see fit. We have purposely left some of the programs in skeletal form so that you can adapt them more easily to your particular taste.



The Program Is the Picture

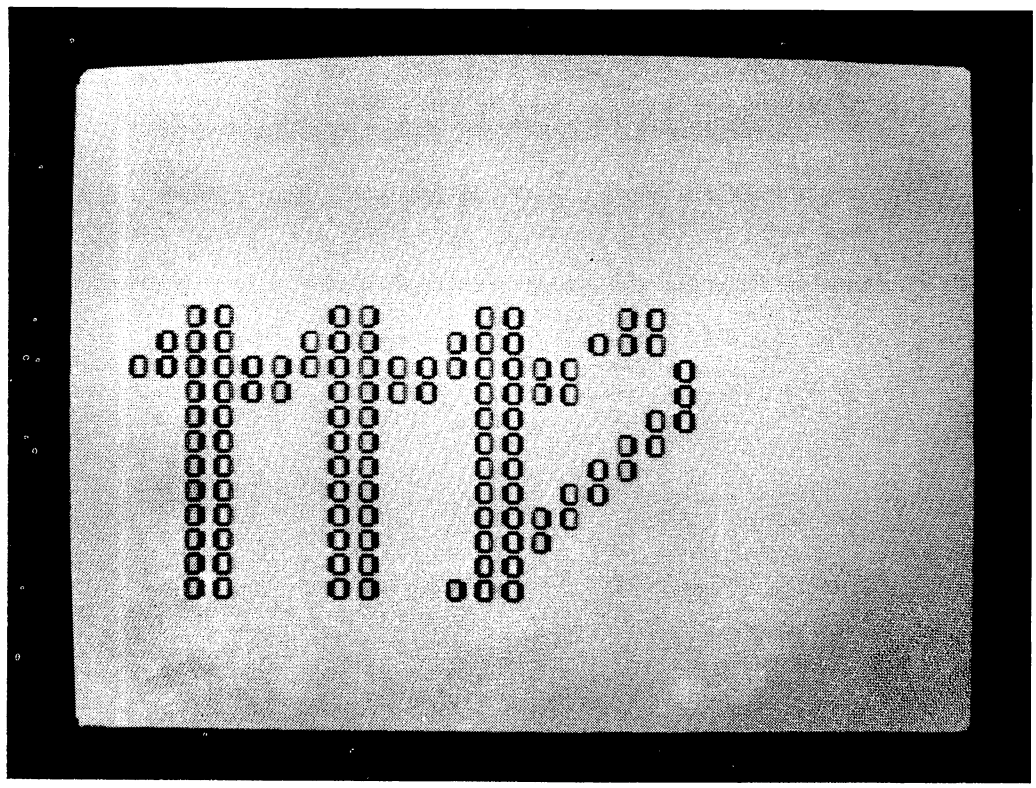
The simplest form of line printer graphics is when the program is the picture. Each program statement is a PRINT statement which outputs a line of the picture. An overwhelming advantage of this technique is that the programmer can detect and correct errors in the picture easily by inspecting a listing of the program.

Problem 1.1

Draw a picture of the Virgo zodiac sign.

Solution

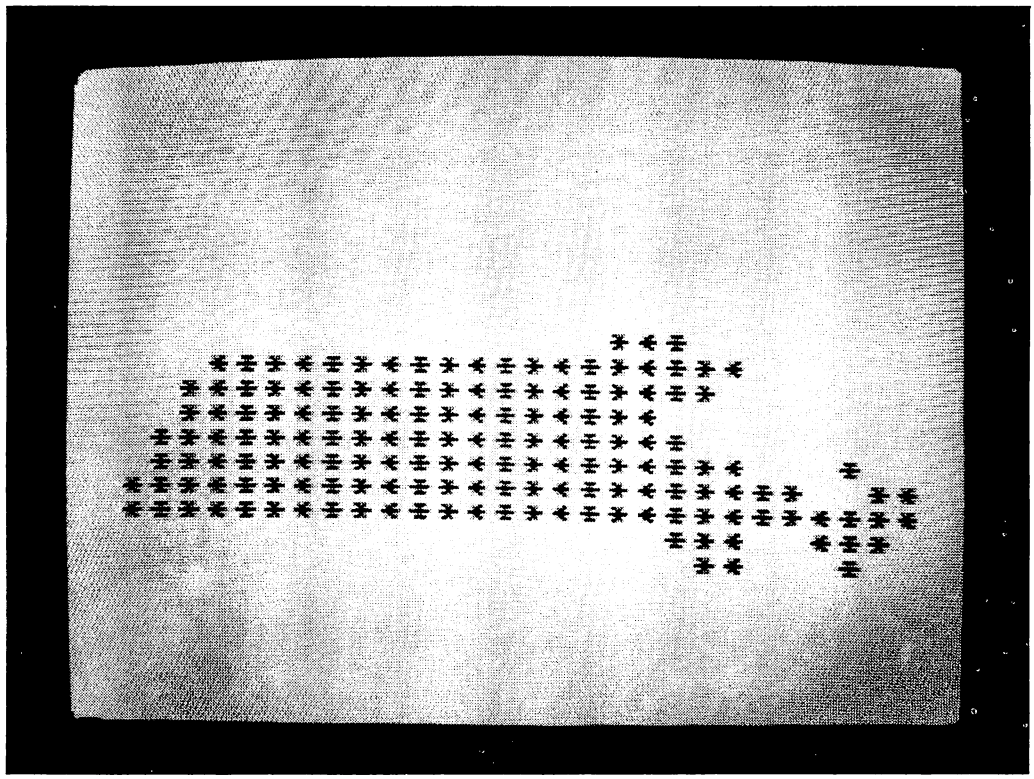
```
10 ! filename:"gr1p1"
20 !  purpose: draw a picture of the Virgo zodiac sign
30 !    author: jpg, jdr & tfz 10/83
40 !
50 PRINT "  00    00    00    00"
60 PRINT " 000   000   000   000"
70 PRINT "0000000000000000    0"
80 PRINT "  0000 0000 0000    0"
90 PRINT "  00    00    00     00"
100 PRINT "  00    00    00    00"
110 PRINT "  00    00    00   00"
120 PRINT "  00    00    00 00"
130 PRINT "  00    00    0000"
140 PRINT "  00    00    000"
150 PRINT "  00    00    00"
160 PRINT "  00    00   000"
170 END
```



Problem 1.2 Draw a picture of the state of Massachusetts.

Solution

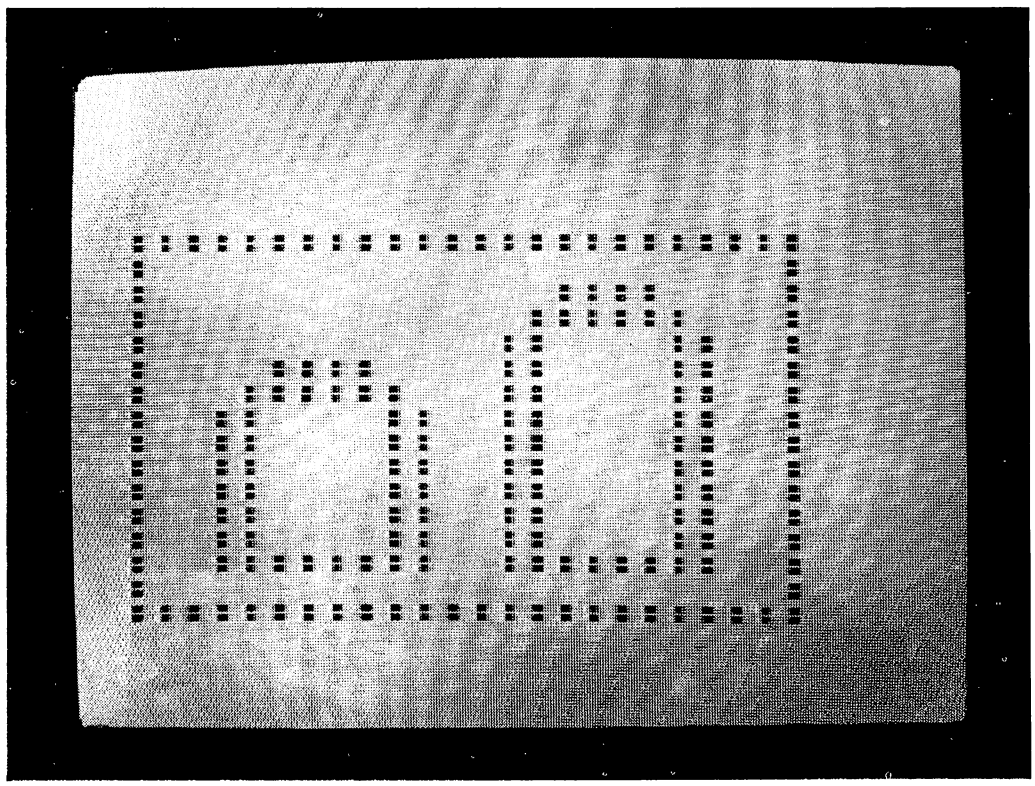
```
10 ! filename:"gr1p2"
20 !  purpose: Massachusetts
30 !   author: jpg, jdr & tfz 10/83
40 !
50 PRINT "                    ***"
60 PRINT "   ******************"
70 PRINT "  *******************"
80 PRINT "  ****************"
90 PRINT " ******************"
100 PRINT " *********************   *"
110 PRINT "************************  **"
120 PRINT "*****************************"
130 PRINT "                     ***  ***"
140 PRINT "                      **   *"
150 END
```



Problem 1.3 Draw the logo for Bentley College.

Solution

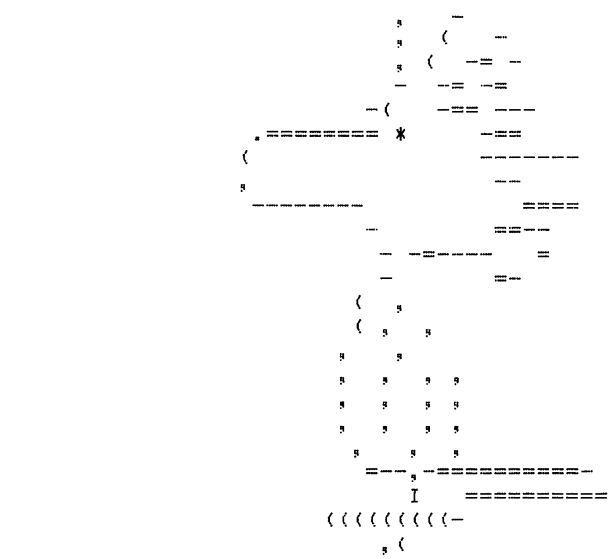
```
10 ! filename:"gr1p3"
20 !  purpose: produce logo of Bentley College
30 !   author: jpg, jdr & tfz 10/83
40 !
50 PRINT ":::::::::::::::::::::::::"
60 PRINT ":                       :"
70 PRINT ":                 ::::  :"
80 PRINT ":               ::::::  :"
90 PRINT ":             ::    ::  :"
100 PRINT ":    ::::    ::    ::  :"
110 PRINT ":  ::::::   ::    ::  :"
120 PRINT ":  ::    ::  ::    ::  :"
130 PRINT ":  ::    ::  ::    ::  :"
140 PRINT ":  ::    ::  ::    ::  :"
150 PRINT ":  ::    ::  ::    ::  :"
160 PRINT ":  ::    ::  ::    ::  :"
170 PRINT ":  ::    ::  ::    ::  :"
180 PRINT ":  ::::::::  ::::::::  :"
190 PRINT ":                       :"
200 PRINT ":::::::::::::::::::::::::"
220 END
```



Problem 1.4 Sketch a picture of Woodstock.

Solution

```
10 ! filename:"gr1p4"
20 !  purpose: draw Woodstock
30 !   author: jpg, jdr & tfz 10/83
40 !
45 OPEN #1:"RS232"
50 PRINT #1:"              ,    -"
60 PRINT #1:"              ,   (    -"
70 PRINT #1:"              , (   -= -"
80 PRINT #1:"              -   -= -=="
90 PRINT #1:"           -(    -== ---"
100 PRINT #1:" .======== *      -=="
110 PRINT #1:"(                 -------"
120 PRINT #1:",                   --"
130 PRINT #1:" --------             ====""
140 PRINT #1:"           -          ==--"
150 PRINT #1:"             - -=====    ="
160 PRINT #1:"             -         =-"
170 PRINT #1:"          (   ,"
180 PRINT #1:"          (  ,  ,"
190 PRINT #1:"         ,    ,"
200 PRINT #1:"         ,  ,   ,  ,"
210 PRINT #1:"         ,  ,   ,  ,"
220 PRINT #1:"         ,  ,   ,  ,"
230 PRINT #1:"          ,    ,   ,"
240 PRINT #1:"           =--,-===========-"
250 PRINT #1:"              I    ==========="
260 PRINT #1:"        (((((((((-"
270 PRINT #1:"             ,("
280 CALL KEY(0,S,N):: IF S=0 THEN 280
290 CLOSE #1
999 END
```



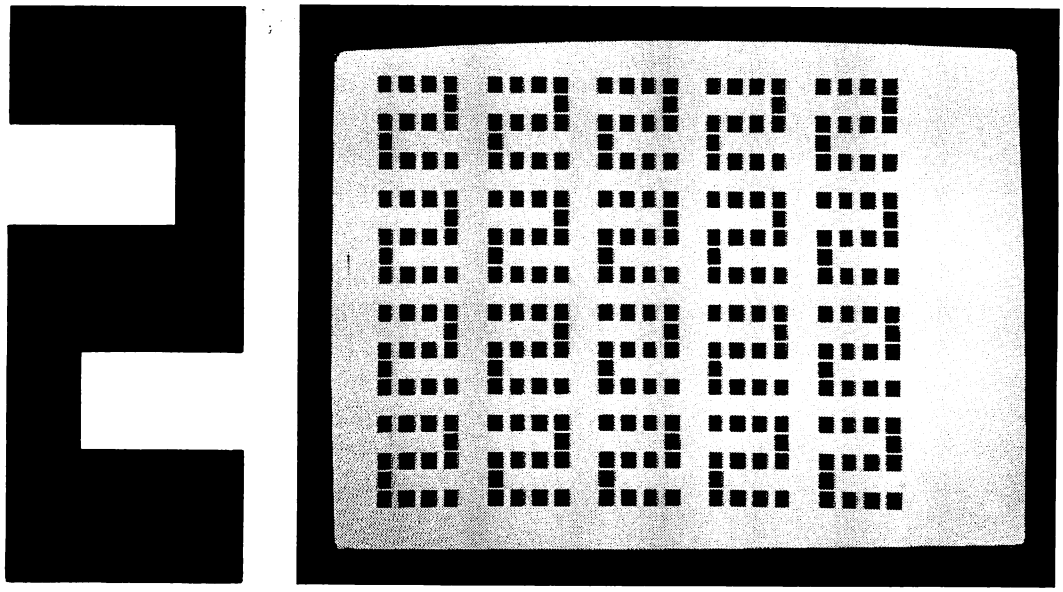
Discussion

- Producing pictures in this way is simple, if you trace either an existing sketch or an original drawing onto a coding form.

- If you use a variety of characters to produce the output, you can more closely approximate the subtle shadings, pleats, highlights, and other details in the picture.

- Woodstock's eye in this display is an asterisk (*). You can change his expression to greed with a dollar sign ($) or open-eyed wonder with a capital O. We leave you to discover how to do this trick. Hint: Read Chapter 6 in this book.

Don't let the simplicity of this graphics technique dissuade you from using it. The very fact that it is so simple makes it immensely popular. Using someone else's artistic ability or your own as a pattern, persistence in the face of tedium coupled with this technique can provide some pleasant computer graphics.



TABbed Pictures

All versions of BASIC have a TAB function that allows positioning of the carriage or print head or cursor anywhere across the output line. Also, the execution of a PRINT statement that doesn't end with a comma or semicolon results in a carriage return-linefeed. With these controls in two dimensions, a wide variety of pictures can be produced. As in the previous chapter, we will illustrate the techniques using both output on a line printer and pictures of actual screen output.

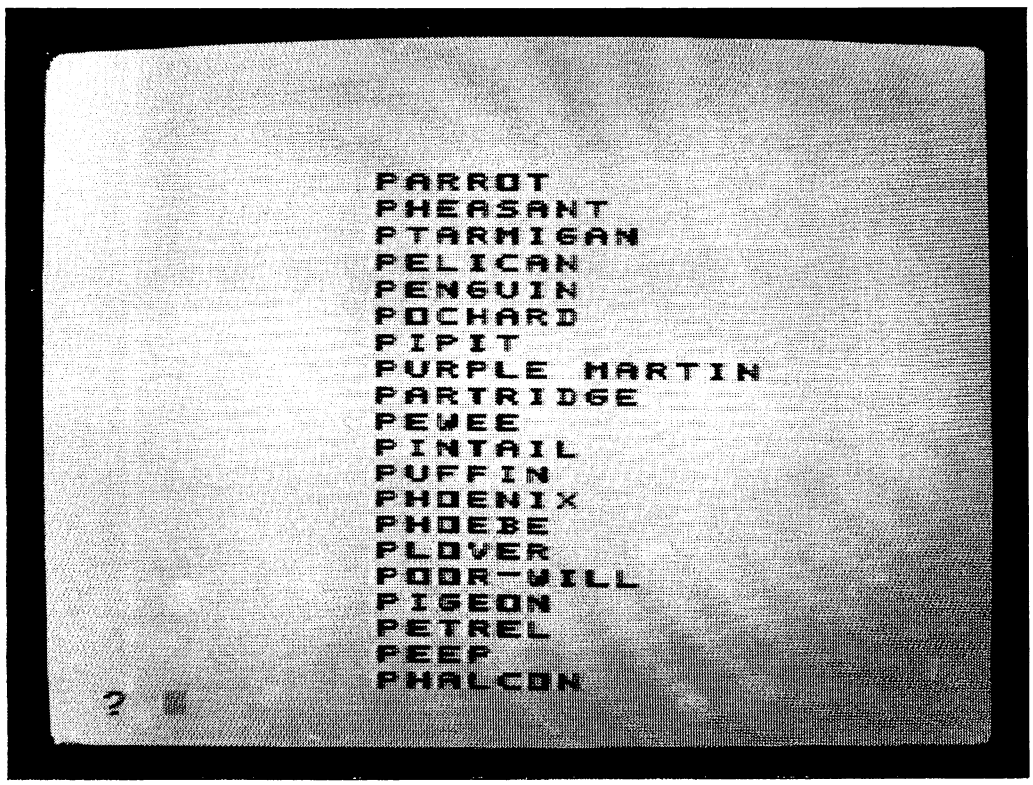
Problem 2.1

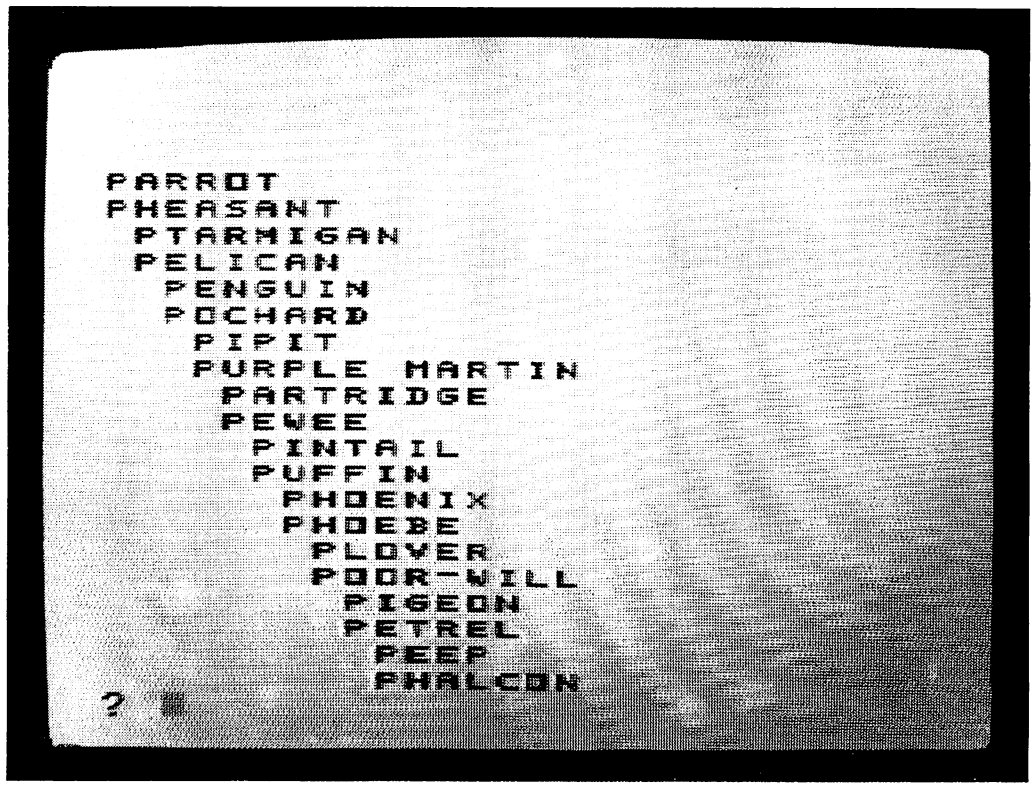
List a table of strings in a variety of ways using the TAB function for positioning the string on the output line.

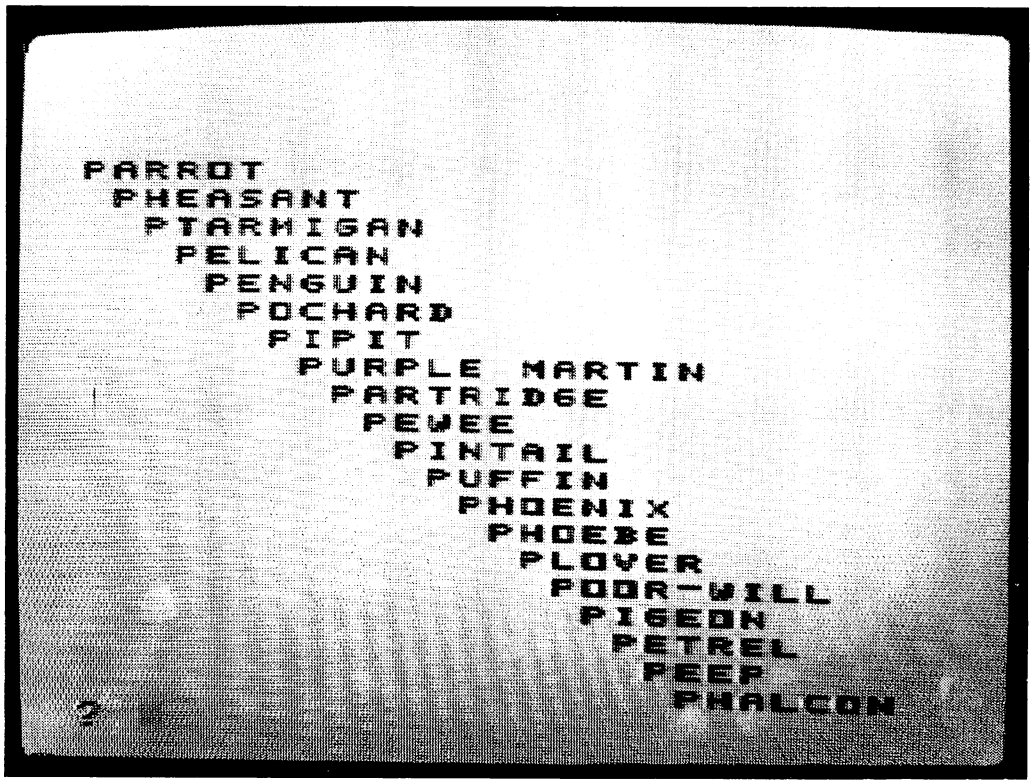
Solution

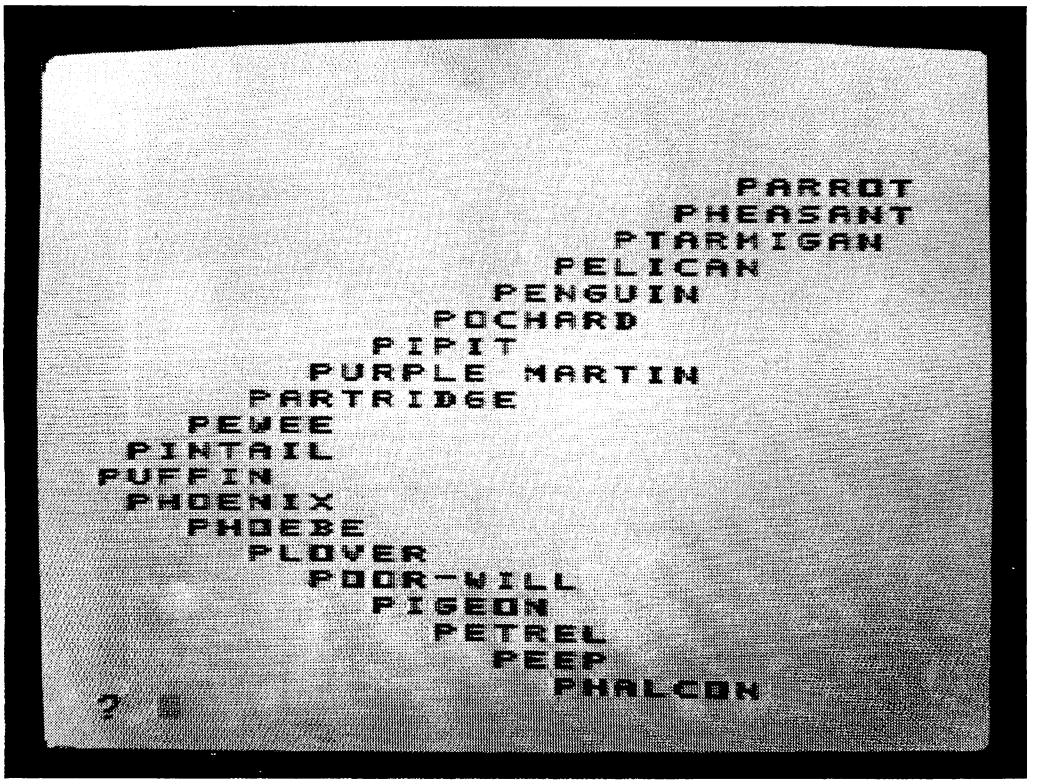
```
10 ! filename:"gr2p1"
20 !  purpose: output table in sundry ways using TAB function
30 !   author: jpg, jdr & tfz 10/83
40 !
50 DIM P$(20)
60 READ N :: FOR I=1 TO N :: READ P$(I):: NEXT I
70 RANDOMIZE :: CALL CLEAR
80 FOR I=1 TO N
90 PRINT TAB(10);P$(I)
100 NEXT I :: GOSUB 1000 :: ! pause and clear screen
110 FOR I=1 TO N
120 PRINT TAB(I);P$(I)
130 NEXT I :: GOSUB 1000 :: ! pause and clear screen
140 FOR I=1 TO N
150 PRINT TAB(.5*I);P$(I)
```

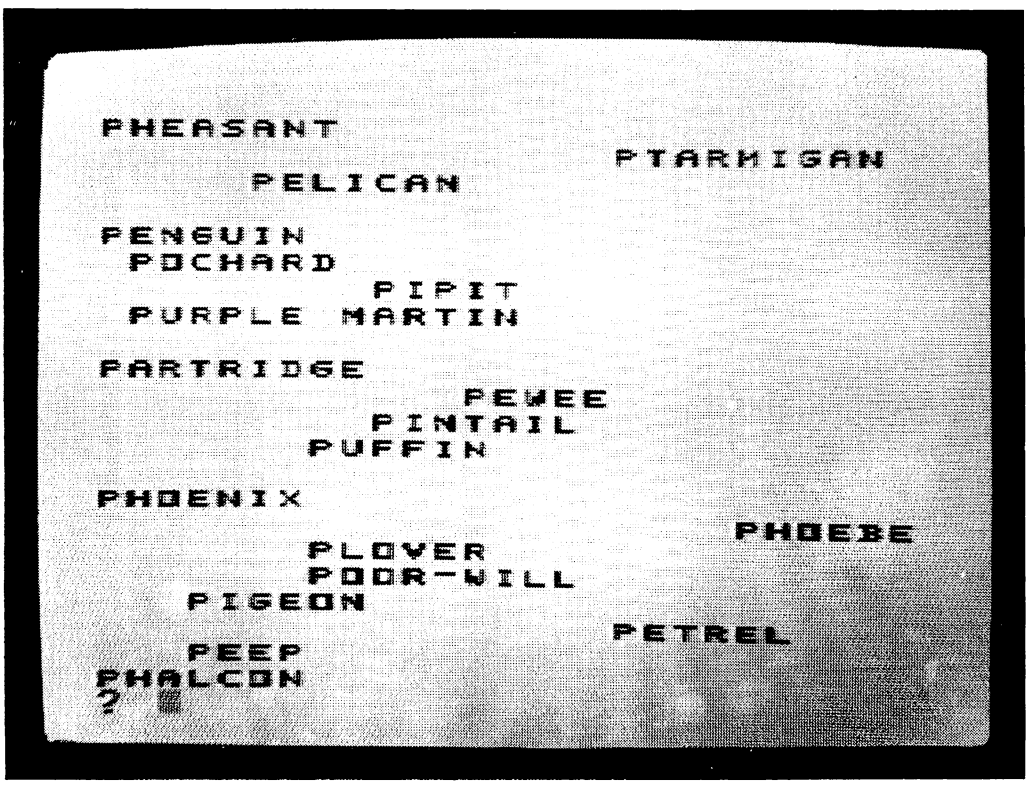
```
160 NEXT I :: GOSUB 1000 :: ! pause ad clear screen
170 FOR I=1 TO N
180 PRINT TAB(2*ABS(12-I));P$(I)
190 NEXT I :: GOSUB 1000 :: ! pause and clear screen
200 FOR I=1 TO N
210 PRINT TAB(1+INT(30*RND));P$(I)
220 NEXT I :: GOSUB 1000 :: ! pause and clear screen
230 FOR I=1 TO N
240 PRINT TAB(20-LEN(P$(I)));P$(I)
250 NEXT I :: GOSUB 1000 :: ! pause and clear screen
260 STOP
270 DATA 20,parrot,pheasant,ptarmigan,pelican,penguin
280 DATA pochard,pipit,purple martin,partridge,pewee
290 DATA pintail,puffin,phoenix,phoebe,plover
300 DATA poor-will,pigeon,petrel,peep,phalcon
1000 ! pause and clear screen subroutine
1010 CALL KEY(0,X,S):: IF S=0 THEN 1010
1020 CALL CLEAR
1030 RETURN
9999 END
```

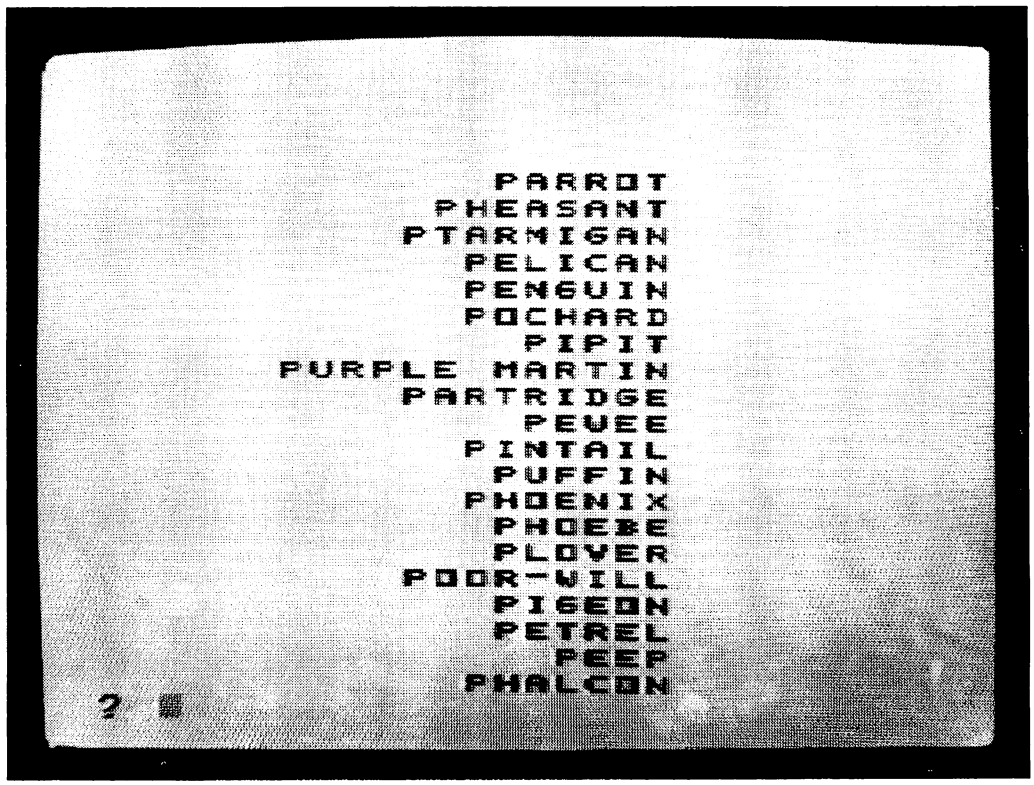



PARROT
PHEASANT
PTARMIGAN
PELICAN
PENGUIN
POCHARD
PIPIT
PURPLE MARTIN
PARTRIDGE
PEWEE
PINTAIL
PUFFIN
PHOENIX
PHOEBE
PLOVER
POOR-WILL
PIGEON
PETREL
PEEP
PHALCON
?


PARROT
PHEASANT
PTARMIGAN
PELICAN
PENGUIN
POCHARD
PIPIT
PURPLE MARTIN
PARTRIDGE
PEWEE
PINTAIL
PUFFIN
PHOENIX
PHOEBE
PLOVER
POOR-WILL
PIGEON
PETREL
PEEP
PHALCON
?


PARROT
PHEASANT
PTARMIGAN
PELICAN
PENGUIN
POCHARD
PIPIT
PURPLE MARTIN
PARTRIDGE
PEWEE
PINTAIL
PUFFIN
PHOENIX
PHOEBE
PLOVER
POOR-WILL
PIGEON
PETREL
PEEP
PHALCON
?


PHEASANT
PTARMIGAN
PELICAN
PENGUIN
POCHARD
PIPIT
PURPLE MARTIN
PARTRIDGE
PEWEE
PINTAIL
PUFFIN
PHOENIX
PHOEBE
PLOVER
POOR-WILL
PIGEON
PETREL
PEEP
PHALCON
?



Discussion

- When the argument of the TAB function is a more complex expression than a constant, some otherwise prosaic output can be transformed into a graphic presentation of information.

Suggestions

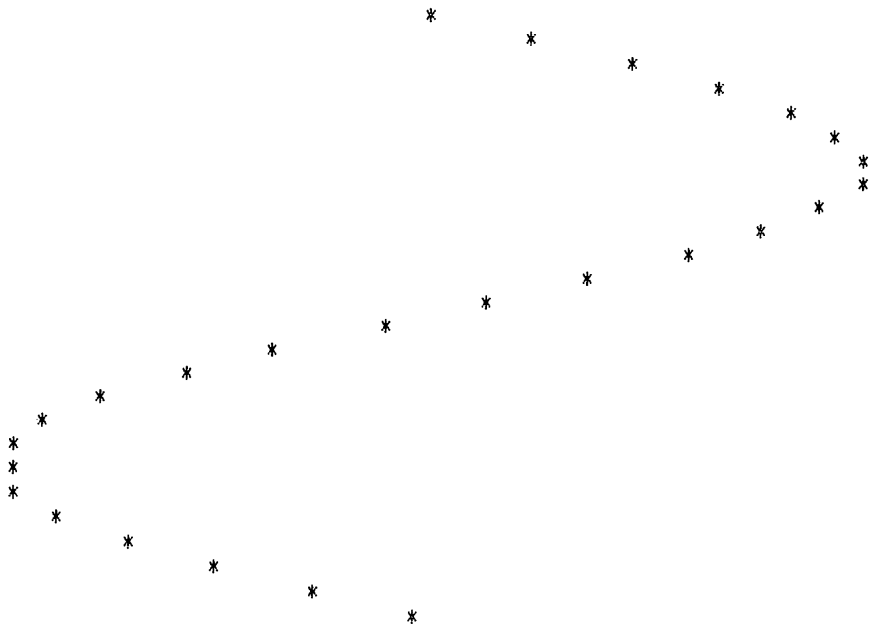
- Modify the argument of the TAB function to produce other interesting visual patterns.

Problem 2.2

Produce a visual representation of the sine curve, y = sin x.

Solution

```
10 ! filename:"gr2p2"
20 !  purpose: graph the sine function
30 !   author: jpg, jdr & tfz 10/83
40 !
50 OPEN #1:"RS232"
60 ! let the X axis vary from 0 to 2 pi radians.
70 FOR X=0 TO 6.28 STEP .25
80 ! calculate displacement from 0 on Y axis,
90 ! then expand by 30.
100 Y=SIN(X)*30
110 PRINT #1:TAB(Y+30);"*"
120 NEXT X
130 CLOSE #1
999 END
```



Discussion

- The Y-axis is across the print line and the X-axis is down the page. Thus, the tabbed position of Y, as represented by the asterisk ("*"), varies according to the sine of the line count X. Change the PRINT #1 to PRINT for screen output.

- The increment between calculated points is .25 radians. To squeeze the sine wave (increase its frequency), increase the increment. To expand the sine wave (decrease its frequency), reduce the increment.

- The sine wave is shown between 0 and 6.28 radians, or 0 to 360 degrees. To see more waves, increment from 0 to 10 (or more) radians. To see the sine function before 0 degrees, increment from -6.28 to 6.28 radians. Note that 6.28 is approximately two times pi, or almost a full circle of 360 degrees.

Suggestions

- Make the following changes in the statements indicated.

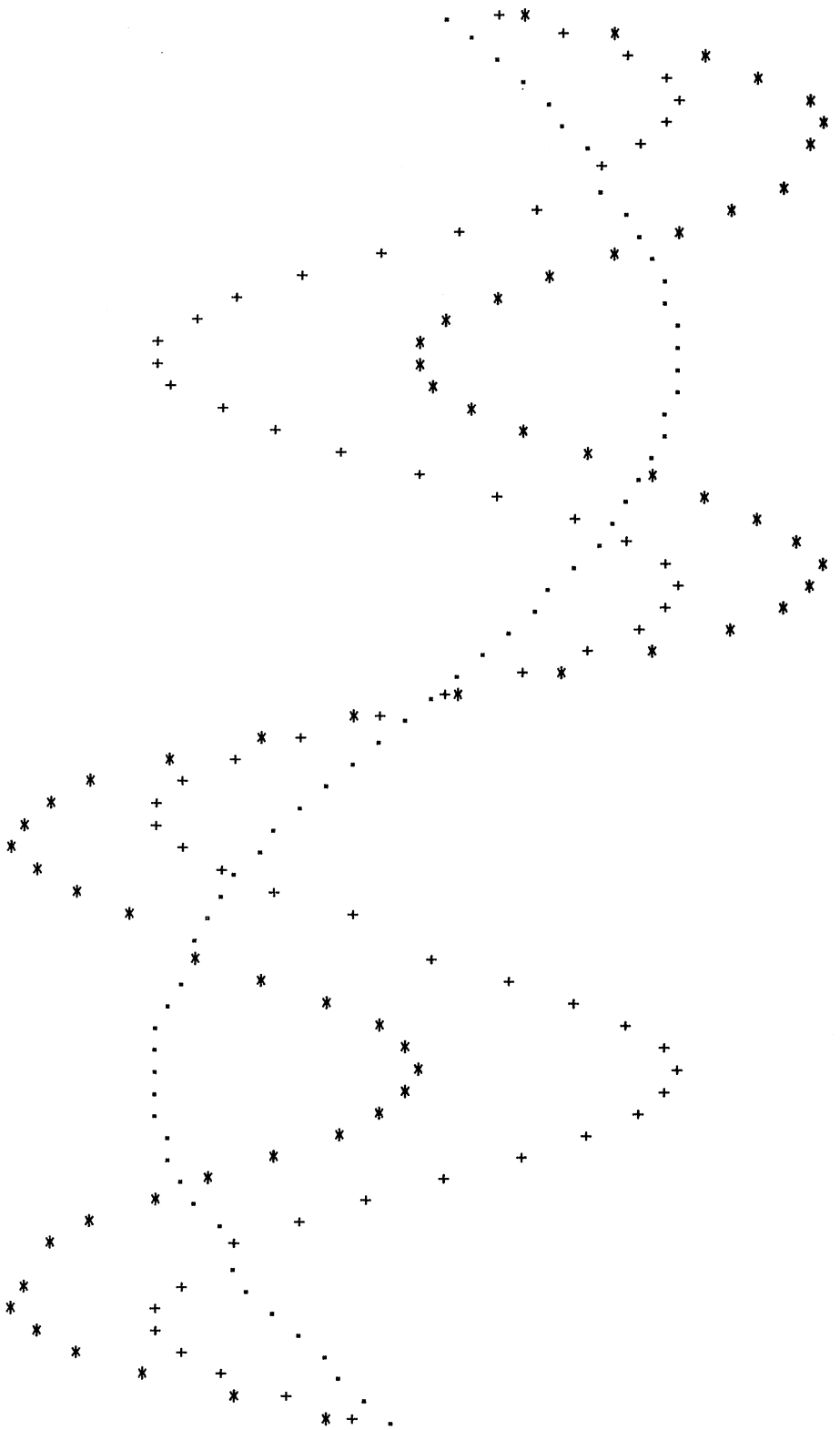
```
60 FOR X=0 TO 6.28 STEP .5
60 FOR X=0 TO 6.28 STEP .1
60 FOR X = -6.28 TO 6.28 STEP .35
60 FOR X= -20 TO 20 STEP .7
90 Y=ABS(30*SIN(X))+32
```

Problem 2.3

Produce a visual representation of three functions simultaneously; a = sin x, b = sin 2x, and c = a + b

Solution

```
10 ! filename:"gr2p3"
20 !  purpose: graph of multiple functions
30 !   author: jpg, jdr & tfz 10/83
40 !
50 OPEN #1:"RS232"
60 ! vary X from .1 to 6.28 radians
70 FOR X=.1 TO 6.28 STEP .1
80 ! set original values of desired functions
90 S=SIN(X):: S2=SIN(3*X):: SS=S+S2
100 ! adjust values of functions for printer
110 A=20*S+32 :: B=20*S2+32 :: C=20*SS+32
120 IF A<=B AND A<=C THEN PRINT #1:TAB(A);".";:: IF B<=C THEN PRINT #1:TAB(B);"+
";TAB(C);"*";ELSE PRINT #1:TAB(C);"*";TAB(B);"+"
130 IF B<=A AND B<=C THEN PRINT #1:TAB(B);"+";:: IF A<=C THEN PRINT #1:TAB(A);".
";TAB(C);"*";ELSE PRINT #1:TAB(C);"*";TAB(A);"."
140 IF C<=A AND C<=B THEN PRINT #1:TAB(C);"*";:: IF A<=B THEN PRINT #1:TAB(A);".
";TAB(B);"+";ELSE PRINT #1:TAB(B);"+";TAB(A);"."
150 NEXT X
160 CLOSE #1
999 END
```



Discussion

- Once you find out which one of the points is lowest in value, print it. Then print the lesser of the remaining two. Then print the one that's left.

- If you graph this kind of multiple function, it is usually necessary to use different characters to plot the different functions.

Suggestions

- Add the first five harmonics of x:
  $y = \sin x + \sin 2x + \sin 3x + \sin 4x + \sin 5x$
  Then, graph y versus x. The result will be a sawtooth wave form.

- Add the first five odd harmonics of x:
  $y = \sin x + \sin 3x + \sin 5x + \sin 7x + \sin 9x$
  Then graph y versus x. The result will be a square wave form.

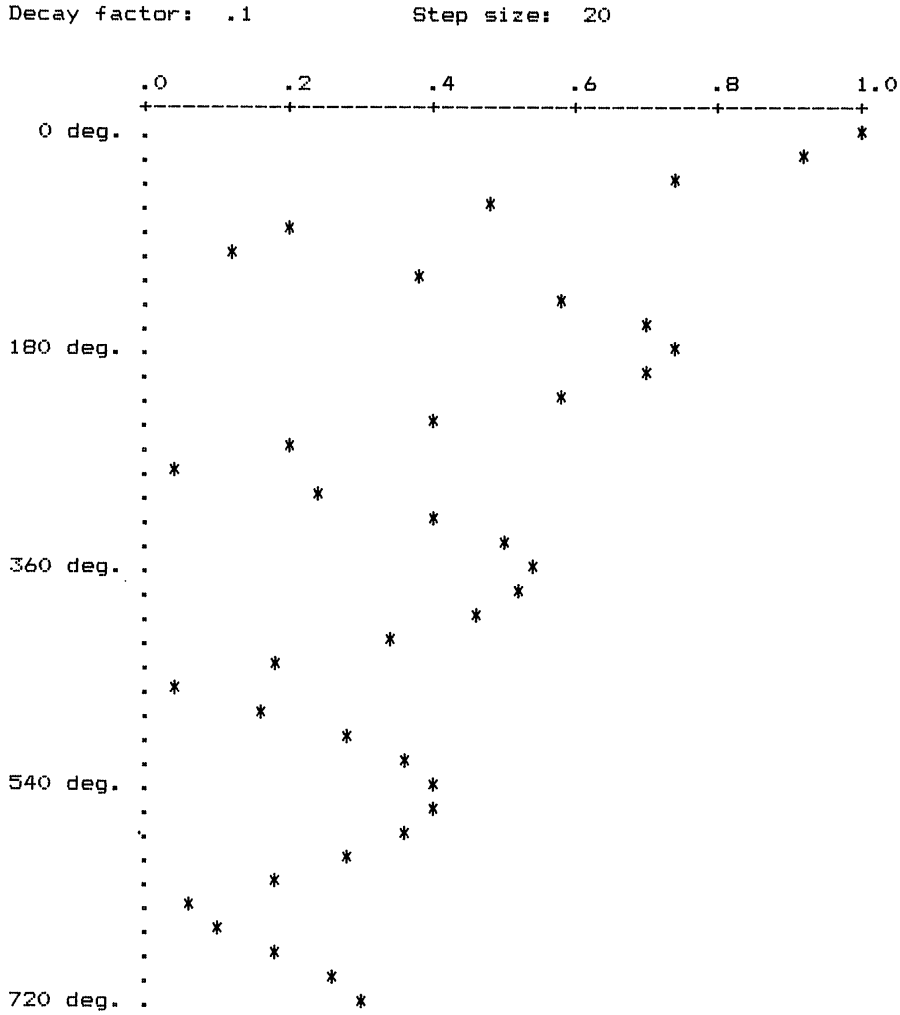
- Graph sin x and cos x together.

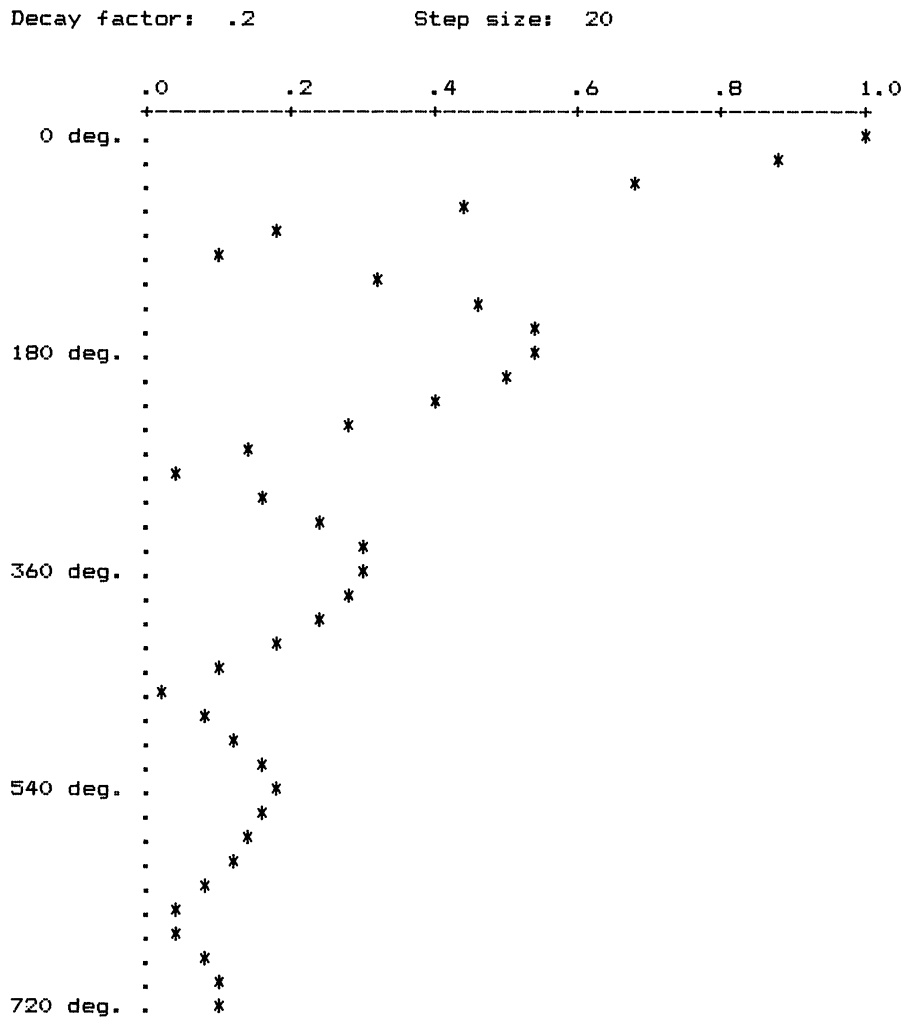
Problem 2.4

Graph a damped cosine curve. More specifically (and complicated sounding), graph two full cycles of a rectified sinusoidal curve that decays exponentially.

Solution

```
10 ! filename:"gr2p4"
20 !  purpose: absolute cosine function decaying exponentially
30 !   author: jpg, jdr & tfz 10/83
40 !
50 INPUT "Enter the decay factor (.01 to .3) ":DF
60 INPUT "Enter the step size in degrees (5 to 30) ":S
70 OPEN #1:"RS232"
80 PRINT #1:"Decay factor: ";DF,"Step size: ";S
90 PRINT #1 :: PRINT #1
100 ! print vertical grid header
110 PRINT #1:TAB(10);".0";TAB(20);".2";TAB(30);".4";TAB(40);".6";TAB(50);".8";TA
B(60);"1.0"
120 ! print vertical grid itself
130 PRINT #1:TAB(10);
140 FOR I=10 TO 60
150 IF I/10=INT(I/10)THEN PRINT #1:"+";ELSE PRINT #1:"-";
160 NEXT I :: PRINT #1
170 ! step from 0 to 720 degrees by steps of S degrees
180 FOR D=0 TO 720 STEP S
190 R=D*.017 :: ! convert degrees to radians
200 Y=49*ABS(EXP(-DF*R)*COS(R))
210 IF D/90=INT(D/90)THEN PRINT #1,USING "### deg.":D;
220 PRINT #1:TAB(10);".";TAB(Y+11);"*"
230 NEXT D
240 CLOSE #1
999 END
```


Decay factor: .1
Step size: 20
.0
.2
.4
.6
.8
1.0
0 deg.
180 deg.
360 deg.
540 deg.
720 deg.


Decay factor: .2
Step size: 20
.0
.2
.4
.6
.8
1.0
0 deg.
180 deg.
360 deg.
540 deg.
720 deg.

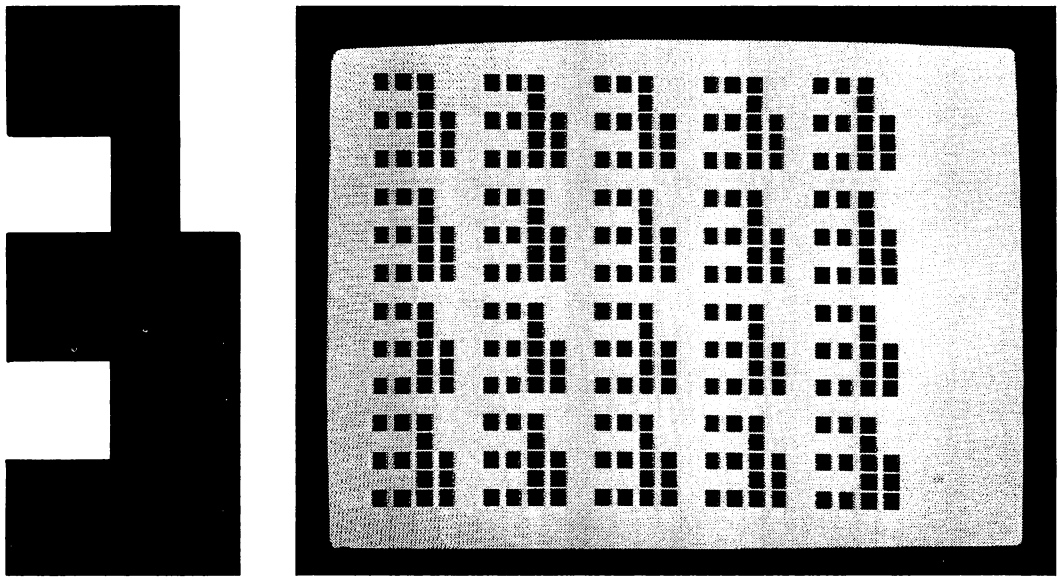
Discussion

- The absolute value function ABS in line 200 does the rectifying by converting all negative values of the cosine function into their positive counterparts.
- The exponential function EXP in line 200 produces the damping effect on the curve.

Suggestions

- Remove the ABS function and shift the graph to the center of the screen: The output is the trace of a moving pendulum as it loses its momentum.
- Modify the program to show two traces at different frequencies.
- Change the EXP function's argument from negative to positive. The effect will be to display an increasing amplitude. This is what happens when an object begins to vibrate at its resonant frequency.

The TAB function can be very useful in the plotting of curves. The examples shown above should provide sufficient patterns for you to explore its use in graphing more exotic or more useful functions.



Bar Graph Pictures

The pictorial representations that are variously called bar graphs or histograms are easy to understand but it can be difficult to get the computer to produce them. If the bars are arranged horizontally, the bar graph's generation is relatively simple to program but the descriptive text is arranged contrary to custom.

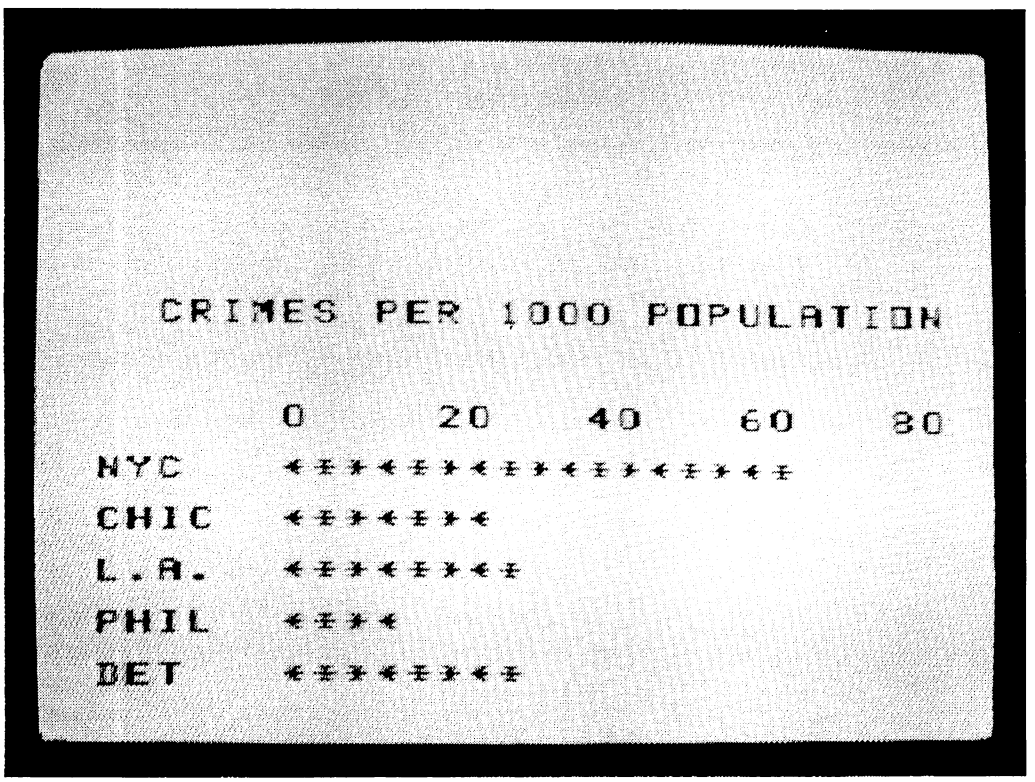
Problem 3.1

Depict the rate of reported crimes in five large metropolitan areas in 1976 using a bar graph.

Metropolitan area	Population, thousands	Crime index, thousands of reported crimes
New York	9,635	658
Chicago	6,982	214
Los Angeles—Long Beach	6,945	247
Philadelphia	4,797	77
Detroit	4,444	154

Solution

```
10 ! filename:"gr3p1"
20 !  purpose: horizontal bar chart
30 !   author: jpg, jdr & tfz 10/83
40 !
50 DIM C$(5),P(5),C(5)
60 FOR I=1 TO 5 :: READ C$(I),P(I),C(I):: NEXT I
70 DATA NYC,9635,658,CHIC,6982,214
80 DATA L.A.,6945,247
90 DATA PHIL,4797,77,DET,4444,154
100 CALL CLEAR
110 PRINT "  CRIMES PER 1000 POPULATION"
120 PRINT :: PRINT :: PRINT
130 FOR I=6 TO 28 STEP 5 :: PRINT TAB(I);(I-6)*4;:: NEXT I :: PRINT
140 FOR I=1 TO 5 :: L=250*C(I)/P(I)
150 PRINT
160 PRINT C$(I);:: GOSUB 190
170 NEXT I
180 CALL KEY(0,N,S):: IF S=0 THEN 180 ELSE 999
190 FOR J=1 TO L :: PRINT TAB(6+J);"*";:: NEXT J :: PRINT :: RETURN
999 END
```



Discussion

- The program graphs the number of reported crimes per ten thousand population.
- Each one of the asterisks on the screen represents one reported crime per 500 population.
- This program could be improved by generalizing it to allow flexibility in defining the tabular data.

Suggestions

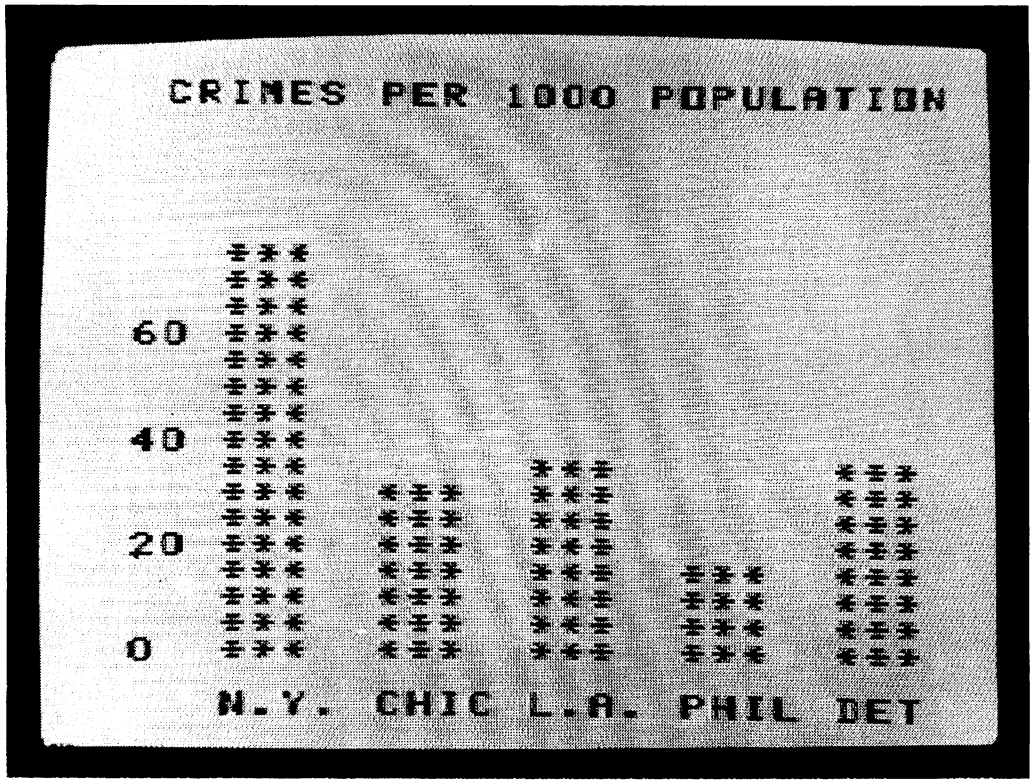
- Provide a dialog before the DIM statement that allows a variable number of cities to be analyzed. This would require the input of the data interactively, rather than through DATA statements.

Problem 3.2

Rotate the bar graph produced in problem 3.1 so that the bars are vertical.

Solution

```
10 ! filename:"gr3p2"
20 !  purpose: vertical bar chart
30 !   author: jpg, jdr & tfz 10/83
40 !
50 DIM C$(5),P(5),C(5)
60 DIM L(5)
70 FOR I=1 TO 5 :: READ C$(I),P(I),C(I):: L(I)=500*C(I)/P(I):: NEXT I
80 DATA N.Y.,9635,658,CHIC,6982,214
90 DATA "L.A.",6945,247
100 DATA PHIL,4797,77,DET,4444,154
110 CALL CLEAR
120 DISPLAY AT(0,3):"CRIMES PER 1000 POPULATION"
130 FOR Y=0 TO 15
140 M=(15-Y)*5
150 IF M/20=INT(M/20)THEN DISPLAY AT(Y+7,0):M
160 NEXT Y
170 FOR Y=0 TO 15
180 FOR I=1 TO 5
190 IF L(I)>(Y+1)*2 THEN DISPLAY AT(22-Y,I*5):"***"
200 NEXT I
210 NEXT Y
220 FOR I=1 TO 5
230 DISPLAY AT(24,I*5):C$(I)
240 NEXT I
250 CALL KEY(0,N,S):: IF S=0 THEN 250
999 END
```



Discussion

- Note the lack of resolution in the heights of the bars. One might assume from looking at this bar graph that Philadelphia's crime rate is two thirds that of Chicago's when in fact it is about half. The flaw lies in the restriction on the height of the bars to just 15 units in order for the bar graph to fit on the screen. This low resolution was not apparent in the previous program because each bar could be as long as 22 characters, yielding almost twice the detail.

Suggestions

- Rewrite the program to accept the data interactively.
- Alter the program to provide more resolution, say bar heights to 30 or 40. Notice that this restricts the program's output to the printer.

Problem 3.3 Write a generalized bar graph generator program.

Solution

```
10 ! filename:"gr3p3"
20 !  purpose: generalized histogram
30 !   author: jpg, jdr & tfz 10/83
40 !
50 DIM B(10),D(500)
60 RANDOMIZE :: CALL CLEAR
70 INPUT "HOW MANY DATA POINTS":N
80 INPUT "WHAT IS LOW VALUE":L :: MN=L
90 INPUT "WHAT IS HIGH VALUE":H :: MX=H
100 OPEN #1:"RS232"
105 PRINT #1:"Num";N,"Low";L,"High";H
110 ! Generate N random data points between MN and MX
120 FOR I=1 TO N
130 D(I)=MN+INT(RND*(MX-MN+1))
140 PRINT #1:D(I);
150 IF I=INT(I/10)*10 THEN PRINT #1
160 NEXT I
170 PRINT #1
180 ! Get height interval size.
190 H=INT(MX-MN+1)/10
200 FOR I=1 TO N
210 ! P is proper bar pointer
220 P=INT((D(I)-MN+1)/H+.5)
230 B(P)=B(P)+1
240 NEXT I
250 PRINT #1
260 FOR I=1 TO 10 :: PRINT #1:B(I);:: NEXT I
270 PRINT #1 :: PRINT #1 :: PRINT #1
280 HT=B(1)
290 FOR J=1 TO 10
300 IF HT<B(J)THEN HT=B(J)
310 NEXT J
320 ! II is vertical interval size (integer)
330 II=INT(HT/10+1)
340 FOR I=10 TO 1 STEP -1
350 PRINT #1:II*I;TAB(5);" ";
360 FOR J=1 TO 10
370 IF B(J)>=I*II THEN PRINT #1:" *** ";ELSE PRINT #1:"-----";
380 NEXT J :: PRINT #1
390 NEXT I
400 PRINT #1:"Size>";
410 T=7
420 FOR I=MN TO MX STEP H
430 K=INT(I+H/2)
440 PRINT #1:TAB(T-1);K;:: T=T+5
450 NEXT I
460 PRINT #1
470 CLOSE #1
999 END
```

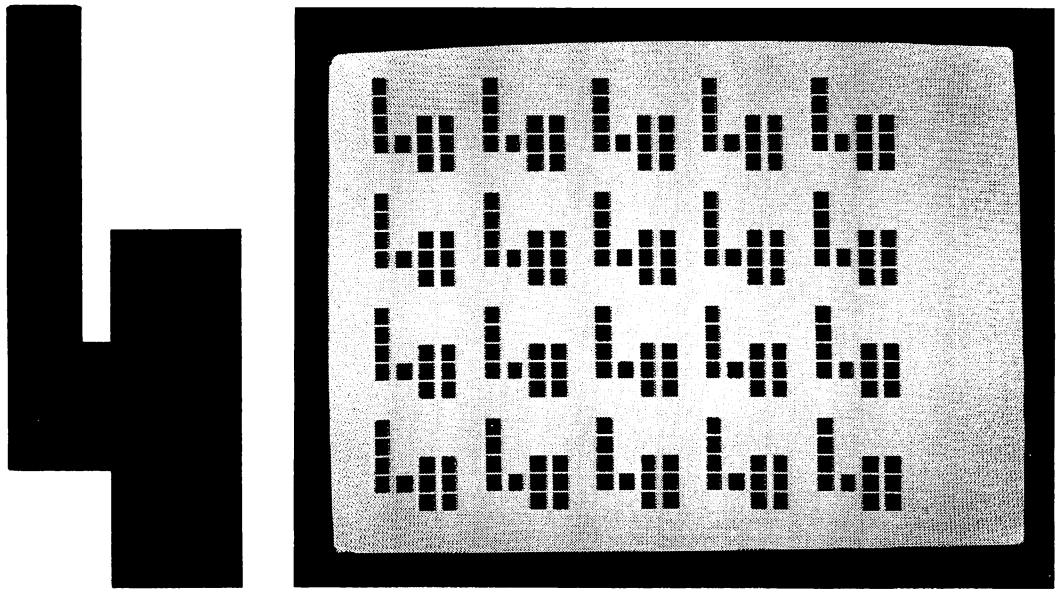
```
Num 200        Low 0          High 99
 51  10  52  92  12   0  91   9  82  29
 58  12  87  44  44  73   3   3  53  76
 31  32  87   6  95  41  98  57   6  17
 58  88  64  16  56   5  87  23  94  37
 32  42  10   1  53  63  82  33  66  65
 61  65  52  26  52  85  79  74   8  84
 46  69  82  70  37  44  87  24  86  18
 99  76  37  74  25  51  63  12  53  16
 44  68  86  39  80  70  22  68  64  31
 82  88  78  16  80  65  75  12  92  24
 60  51  14  66   5  37  46  52  63  70
 15  24   7  30  98  16  53  43   9  16
 55  31  86  93  67  63  88  75  76  63
 27  51   6  76  80  53  65  44  44  21
 42  66  16  35  19  77   5  63  46  95
 10  12  61  23  47  50   0  89  11  80
 47   2  64  68  61  63  33   5  15   3
 23  80  53  34  69  54   3  26  96  87
 89  59  94  23  77  47  22   1  79  87
 71  15  87  76  20  27  36  47  98  67


 20  21  17  12  28  18  22  23  21   9


 30  ----------------------------------------------------
 27  -------------------- *** ------------------------
 24  -------------------- *** ------------------------
 21  ----- *** ---------- *** ----- *** *** *** -----
 18   ***  *** ---------- *** *** *** *** *** -----
 15   ***  ***  *** ----- *** *** *** *** *** -----
 12   ***  ***  ***  ***  *** *** *** *** *** -----
  9   ***  ***  ***  ***  *** *** *** *** ***  ***
  6   ***  ***  ***  ***  *** *** *** *** ***  ***
  3   ***  ***  ***  ***  *** *** *** *** ***  ***
Size> 5    15   25   35   45   55   65   75   85   95
```

Discussion

- This is a generalized bar graph generator which can apply to most kinds of data. It provides for 10 bars oriented vertically.
- Note that the program's calculations are almost entirely for purposes of scaling the vertical and horizontal dimensions of the graph.
- The data values that are graphed are produced by the program itself in lines 120-160. Of course this section of the program should be rewritten to allow the user to either enter the data conversationally or provide DATA statements which the program would READ.

Suggestions

- Rewrite the program to have the user provide the data.
- Modify the program to give the user the choice of symbol used for printing the bars.
- Rewrite the program to give the user the choice of screen or printer output.



Computed Pictures

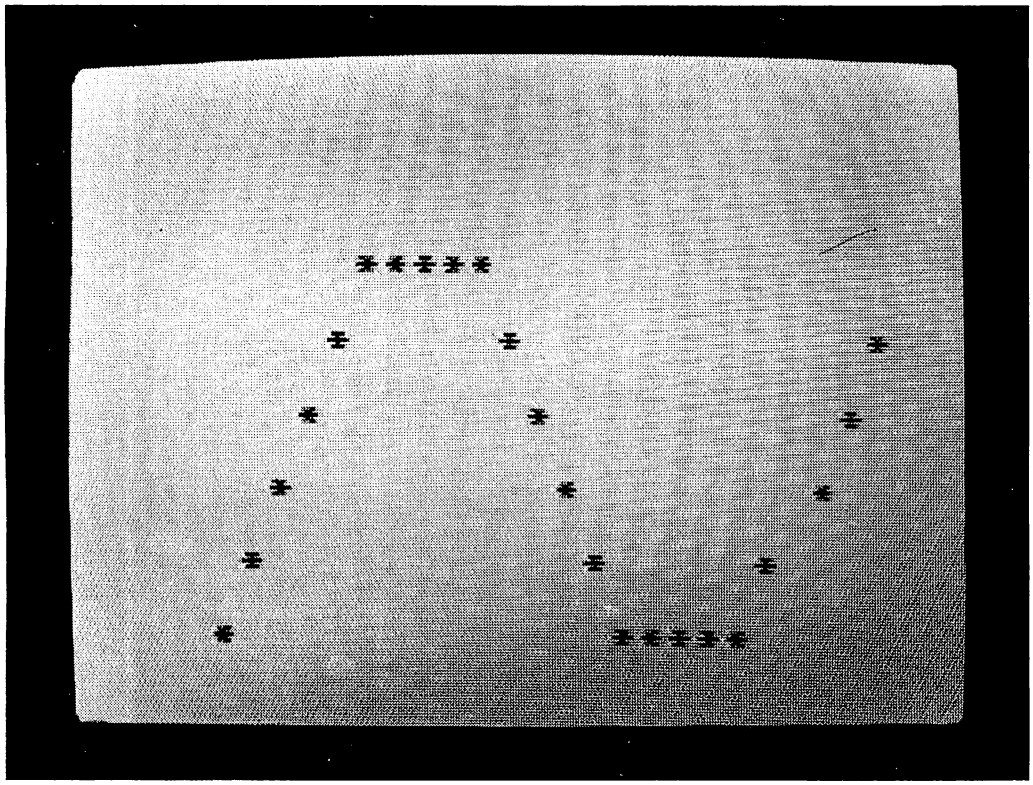
This chapter illustrates a technique that uses a two-dimensional table to hold numeric values that have been computed according to the solution of some problem. This table is then printed with the conversion of the numeric table elements to a symbol that provides a meaningful "picture" of the computation.

Problem 4.1

Rotate the graph of the sine curve of problem 2.2 so that the X-axis is horizontal and progresses across the screen.

Solution

```
10 ! filename:"gr4p1"
20 !  purpose: sine wave across screen
30 !   author: jpg, jdr & tfz 10/83
40 !
50 CALL CLEAR :: DIM P(80,25)
60 FOR A=0 TO 8 STEP .35
70 X=INT(A*3+4):: Y=INT(SIN(A-1)*3)+4 :: P(X,Y)=1
80 NEXT A
90 FOR Y=25 TO 1 STEP -1 :: X$=""
100 FOR X=1 TO 80
110 IF P(X,Y)=1 THEN X$=X$&"*" ELSE X$=X$&" "
120 NEXT X
130 PRINT X$;
140 NEXT Y
150 CALL KEY(0,N,S):: IF S=0 THEN 150
999 END
```



Discussion

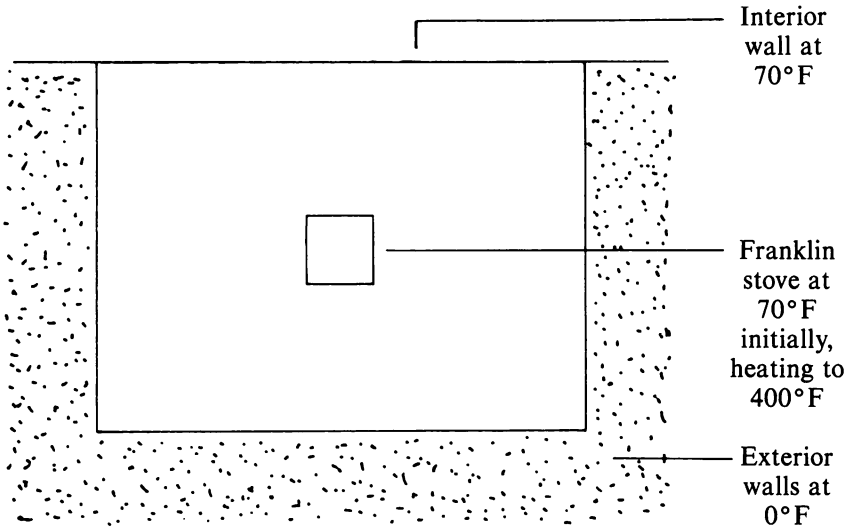
- A lot of memory is used to store the table. Each position of the table is an integer value that indicates whether or not that position is on the sine curve.
- Each line of output is produced directly by building a string of mostly blanks and a few asterisks "(*)" from scanning a row of the table.
- The program uses the fact that BASIC sets all variables to 0 initially. This is not a good programming practice, because if the program was to be used as a subroutine and was called more than once, it would "remember" all elements of the table that had been set previously to 1. To make the program more generally useful, you should set each element of the table to zero before each use.

Suggestions

- Try modifying the program so that a composite graph of sines and cosines can be made. You'll have to devise a way to encode three potential symbols; point for sine, point for cosine, and no point at all. What if the sine and cosine intersect? Maybe a fourth symbol could be used to show this occurrence.
- Cause the X-axis and Y-axis to be printed along with the graphs.

Problem 4.2

Display the changing thermal gradient around a rectangular fireplace located at the center of a rectangular room. Imagine a room surrounded by exterior walls on three sides with a Franklin stove in its center. It's a cold, overcast winter day. What is the heat distribution in the room as the exterior of the Franklin stove begins to heat from 70°F to 400°F?



The problem resembles a problem on page 98 of Daniel D. McCracken's *A Guide to FORTRAN Programming*, (Wiley, 1965).

Solution

- Consider the room to be 40 feet long and 30 feet wide. The exterior walls are at a constant 0°F and the interior wall is a constant 70°F.

- Reserve a 30 x 40 integer table X in memory, with X(0,1) to X(0,39) held at a value of 70; X(30,0) to X(30,40), X(0,0) to X(0,30), and X(0,40) to X(30,40) held at a value of 0. These are the walls of the room. The 6-foot wide by 4-foot deep center area is the Franklin stove that starts at 70°F and ends at 400°F.

- Proceed to compute the temperatures between wall and stove, from X(1,1) to X(1,39), then X(2,1) to X(2,39), . . . , through X(29,1) to X(29,39), calculating each point on the basis of its neighbors according to the formula:

  $$X(I,J) = (X(I-1),J) + X(I,J-1) + X(I+1,J) + X(I,J+1))/4$$

  This formula calculates the temperature of a point by averaging the temperatures of that point's four neighbors.

- Repeat the computations for as many iterations as the user wishes.

- Convert all numeric values of table X to characters one 64-character line at a time using the chart of symbols below to indicate various temperatures.

Integer value	Symbol
0- 9	a
20- 29	b
40- 49	c
.	.
.	.
.	.
380-389	u
400-409	v

  All temperatures in unspecified intervals like 10-19, 30-39, etc., are symbolized with a blank.

- Paint the picture on the screen or line printer one line at a time.

- Repeat the above process increasing the temperature of the Franklin stove by 50 degrees, until it reaches 400 degrees.

- Continue the above for as many iterations as the user wishes or until the output is a picture of the room at equilibrium conditions.

```
10 ! filename:"gr4p2"
20 !  purpose: graph thermal gradient in a heated room
30 !   author: jpg, jdr & tfz 10/83
40 !
60 ! set up initial conditions
70 D=30 :: W=40 :: ! room measurements
80 LL=5/12*W :: LR=7/12*W :: ! stove position
90 LT=5/12*D :: LB=7/12*D
100 INPUT "INNER, OUTER WALL TEMPERATURES ":IN,OU
110 INPUT "STARTING, LAST, STEP TEMPERATURES OF STOVE ":ST,FI,CY
120 INPUT "TOTAL ITERATIONS, NUMBER PER PRINTING CYCLE ":IT,Q
130 DIM X(30,40):: DIM S$(52):: !room measurements
135 OPEN #1:"RS232"
140 PRINT #1:"INNER=";IN,"OUTER=";OU,"STOVE=";ST;"TO";FI;"BY";CY
150 PRINT #1:"ITERATIONS=";IT,"NUMBER/PRINTING CYCLE=";Q
160 FOR I=0 TO 51 STEP 2 :: READ S$(I):: NEXT I
170 DATA a,b,c,d,e,f,g,h,i,j,k
180 DATA l,m,n,o,p,q,r,s,t,u,v,w,x,y,z
190 FOR I=1 TO 52 STEP 2 :: S$(I)=" " :: NEXT I
200 ! set initial room temperature to random between 0 and 400
210 FOR I=1 TO D-1 :: FOR J=1 TO W-1
220 X(I,J)=INT(RND*40)
230 NEXT J :: NEXT I
240 ! set inner wall,bottom outer wall to starting temperatures
250 FOR I=0 TO W :: X(0,I)=IN/10 :: X(D,I)=OU/10 :: NEXT I
260 ! set left and right outer walls to starting temperatures
270 FOR I=0 TO D :: X(I,0)=OU/10 :: X(I,W)=OU/10 :: NEXT I
280 ! set stove to starting temperature for this cycle
290 FOR S=ST/10 TO FI/10 STEP CY/10
300 PRINT #1 :: PRINT #1:"STOVE IS AT";S*10;"DEGREES"
310 FOR I=LT TO LB :: FOR J=LL TO LR :: X(I,J)=S :: NEXT J :: NEXT I
320 CALL CLEAR
330 FOR N=1 TO IT :: I=0 :: GOSUB 1000
340 FOR I=1 TO D-1
350 ! weave calculations back and forth
360 ! instead of always left to right
370 IF I/2=INT(I/2)THEN S1=1 :: S2=W-1 :: S3=1 ELSE S1=W-1 :: S2=1 :: S3=-1
380 FOR J=S1 TO S2 STEP S3
390 ! calculate all inner values.  if stove, bypass
400 IF I>=LT AND I<=LB AND J>=LL AND J<=LR THEN 420
410 X(I,J)=(X(I-1,J)+X(I,J-1)+X(I+1,J)+X(I,J+1))/4
420 NEXT J :: GOSUB 1000
430 NEXT I :: GOSUB 1000
440 NEXT N
450 NEXT S
460 CALL KEY(0,N,S):: IF S=0 THEN 460
470 STOP
1000 ! subroutine to graph a single line
1010 A$="" :: FOR K=0 TO W
1020 ! the symbol # is reserved for the stove's position
1030 IF K>=LL AND K<=LR AND I>=LT AND I<=LB THEN A$=A$&"#" ELSE A$=A$&S$(X(I,K))
1040 NEXT K
1050 PRINT A$,N;I
1060 IF INT(N/Q)=N/Q THEN PRINT #1:A$,N;I
1070 RETURN
1080 CLOSE #1
9999 END
```

```
INNER= 70      OUTER= 0        STOVE= 70 TO 400 BY 50
ITERATIONS= 9                  NUMBER/PRINTING CYCLE= 3

STOVE IS AT 70 DEGREES
a                                                  a  3  0
a gh hhh  gg h i hh g f   gg   g  hh  gea  3  1
a       j i ijkk  ii h hhhh   i  j k   ea  3  2
af  l ll kjjj   kj  j   i h i   kk  ml fa  3  3
a jlmmm lk j k k j jjjjjjiihi jkk l ml  a  3  4
afjlmmm lkjjjk k jj   jjjj   jj   k     a  3  5
af k l l kji jkk  k k     j   jj  k jig a  3  6
a  j  l l ji   kk ll  iiii kk   k kji fda  3  7
ae i     lk i  jkl l   iij k   kl lkj  da  3  8
ae   jk    kj i   l  k   j kkkkkl m k gda  3  9
a ghijkll k jj  jkll kkk k k    llm  j  a  3  10
a ghi kllk jjjjj   kk j k    l kkk lkj  a  3  11
a g i kll k jj j hhhhhh jkkk kk  kkkk gda  3  12
ae i j mm  k    h########hj   kkk  kk i da  3  13
aehijk mm    kk  ########  jj k k  kkjigda  3  14
a  j kl   lllk j ########  jj   k   jj   a  3  15
a   j   llll k jh########h  j   kk j  j ea  3  16
a  jj kk lllk  jh########hj j  l kk     a  3  17
a   j   k l k    gg  ggh jj kll kkk kjhea  3  18
a  j k     kk kk i     ij  k l   kk    ea  3  19
a hjkk       kkkj  jjj    k  ll l k  i ea  3  20
aeh jjjjj  k kk j j kk kk  k  llll k  hea  3  21
ae  i  ij kkkkk   j   k ll  k l  llk jhea  3  22
a g  hh  j kkkkk  j jj   ll   ll llkj hea  3  23
a  iiiii  j   k     jj   k lll   l kj   a  3  24
aehj j i i  j k  jjjj  jj k l  kkkk  i ea  3  25
ae j  ji   j  kk jjj kk jj kkk   jj i   a  3  26
a  i j i hi     k jj kk jjj      ji h   a  3  27
ad   h  g h  iii iiii ii iiiiiiii  g feca  3  28
a  d      e                    eee d cba  3  29
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  3  30
```

Discussion

- The previous output shows the effect of starting the room's temperature at 0 degrees, the same temperature as the exterior walls. We arbitrarily started the interior of the room at various random settings between 0 and the temperature of the stove, and while not realistic, it seems to be most effective in reaching equilibrium quickly.

- The program was limited to this size room for practical reasons of time. It takes the TI–99/4A about 2 minutes to proceed from one graph to the next, with an interval size of 5 iterations.

```
INNER= 70      OUTER= 0       STOVE= 70 TO 400 BY 50
ITERATIONS= 9                 NUMBER/PRINTING CYCLE= 3

STOVE IS AT 70 DEGREES
a                                          a  3  0
ae          g     h  ggg h          ggg  hgea  3  1
ae   i j ii ii j iiii j    j  h h       i a  3  2
ae i j  k jjjj   jj  j k     ji i jj j jga  3  3
ae      l      k kk j j        j  j k  kkjfa  3  4
a g     l     k l k  j k     j  j j  kkkkifa  3  5
a  hijk kk k     k  k  k j kk     jjj  ki a  3  6
ad  h jkkkkkkkkkkkkkk  k kk    j i   kk fa  3  7
a g h  j  kkjj   l kkk k k   k  iij    fa  3  8
a h   i  j  jj klll kk kkkk  l     j    fa  3  9
a h iiii j  jjjkkkkkkkk      lllkj jkk   a  3  10
a  ii      j  j i     jjj   jk       j k i a  3  11
a  hi      kkk ji          ij kl l j  lki a  3  12
ad h   ll  ll kjh########g  i  l k   l i a  3  13
ad iklm mmmmm   h########  hhikkk  k  kiea  3  14
a  j  m m l   l h######## hhhh  kk kk   fa  3  15
aeh klm l kkk   h########     ijjkkk k  hea  3  16
ae  jkll k jj k  ######## h jjj   k k igea  3  17
ae i  k k jjjjkkjhg ffff   j     j  k ig a  3  18
ae  jj kkjjjj  k   hg g hi j jj  i       a  3  19
a   jk k  j  kkk   ii  h i jj i h i jjhea  3  20
a i kkk kkkkkkk jj k j i i jj  hh  i    a  3  21
af j    k lll  jj k     iij    h i jj   a  3  22
aehj   k llll      k    j    k i  jj jjh a  3  23
a h jjk    kkk    j   k j  llkj j k  j ea  3  24
ae i j k kk    j i  j kkkk l     j jjjj fa  3  25
aehi jjkkk  jjj  j     lll l  k j      fa  3  26
aeh j  jjjj  i  j j    ll k k  kj  j  i a  3  27
a ghhhhhhhhh   h  hh ijj i  ijj i  i  hea  3  28
a deeeeeeeee    eeee   ff eee ff      e a  3  29
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  3  30
```

Suggestions

- Use other symbols to represent temperature, such as:

Temperature	Symbol
0- 9	blank
10- 19	.
20- 29	:
30- 39	;
40- 49	-
50- 59	=
60- 69	1
70- 79	I
80- 89	L
90- 99	T
.	.
.	.
.	.
400-409	#

A minor modification of this program can yield startlingly different results. The modification is based on doing an exact point-by-point calculation of a location's value. This kind of calculation isn't appropriate for thermal gradients, as was the case with the wood-burning stove, but it works for a wide variety of applications in which point values are based on field effects, such as electricity, gravitation, and sound.

Consider a "generator" of such a field located at a point in the middle of a rectangularly shaped area. The field strength at any other point is proportional to the square of the distance away from the generator. Thus a sound is one fourth the volume at 20 feet from a speaker as it is 10 feet away. If there are two speakers located in the rectangular area, the intensity of the sound at any point is the sum of the sound from the two speakers. It is easy to determine the distance to each of the speakers by use of the Pythagorean theorem ($c^2 = a^2 + b^2$). Then the process of summing the individual intensities, taking into account their distances and original intensities, yields the total amount of sound at a selected point. Repeat this process for all points on the grid, and represent point intensities by graphical means. The effect is a "picture" of the sound sources and their acoustic "fields".

The same process can be used to picture any physical field phenomenon, such as gravitational "wells" around planets and ionic field strengths.

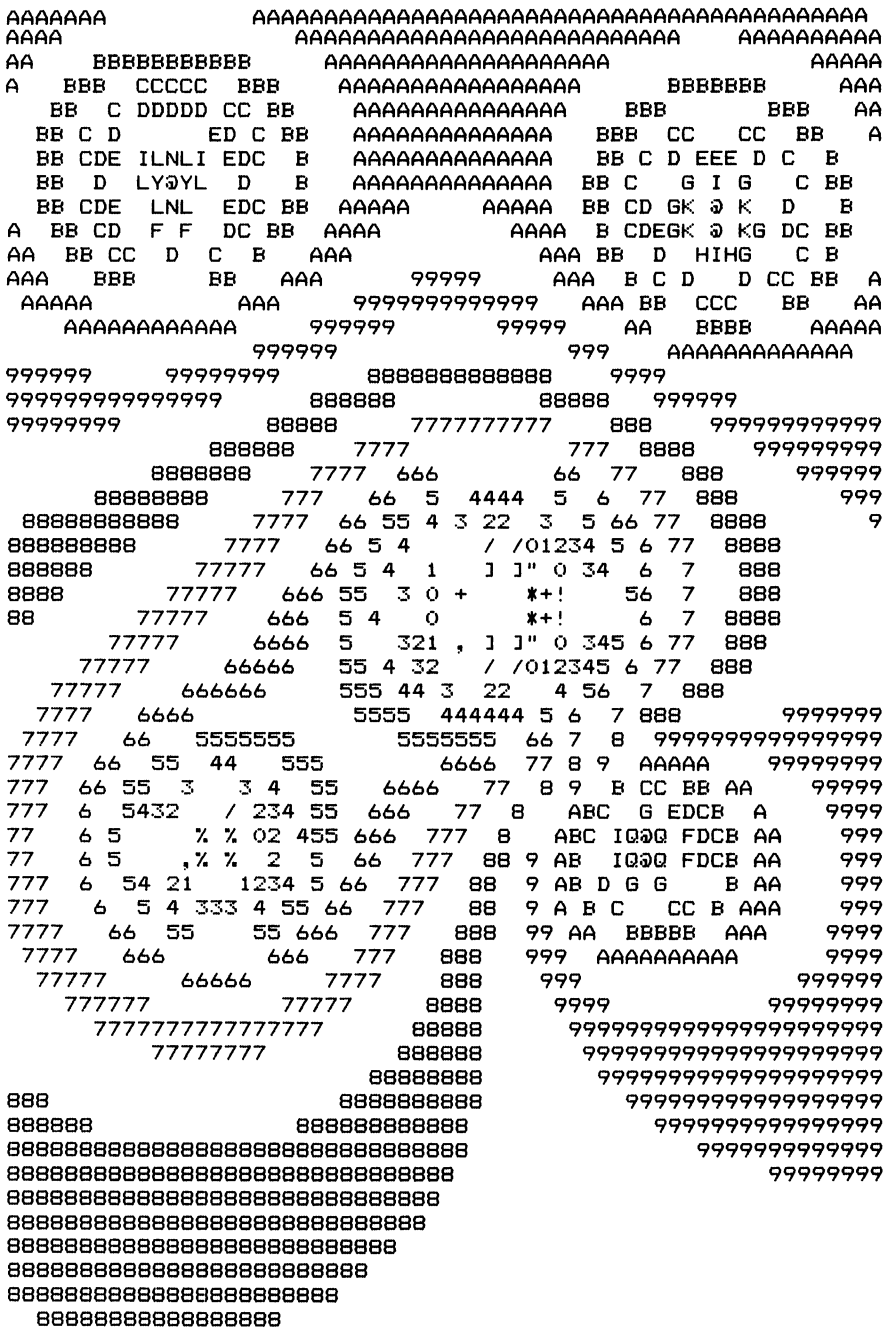
An excellent discussion and example of this method of graphing can be found in *BASIC and the Personal Computer* by Dwyer and Critchfield, (Addison-Wesley, 1978), pages 304-306.

Problem 4.3

Display the point-by-point ionic field strength that surrounds each of up to five electrical sources. Each source can be positive or negative in relation to the background field strength. Each source will have a strength from −2 to +2. Pictorially, a negative field will be represented as a series of digits and characters ranging from 9 through 0 to such characters as /, !, +, *, . . ., %, and @. A positive field will be represented as a series of letters, from A to Z. Field areas closest to background will be represented as 9s or As.

Solution

```
10 ! filename:"gr4p3"
20 !  purpose: to graph electrical charges
30 !   author: jpg, jdr & tfz 10/83
40 !
50 DIM A$(52),A(26),B(5),C(5)
60 OPEN #1:"RS232"
70 DATA 35,64,36,42,38,37,94,91,93,95,43,34,40,44,33
80 DATA 47,48,49,50,51,52,53,54,55,56,57
90 FOR I=1 TO 26 :: READ A(I):: A$(I)=CHR$(A(I)):: NEXT I
100 FOR I=27 TO 52 :: A$(I)=CHR$(I+38):: NEXT I
110 N=5 :: A(1)=15 :: B(1)=12 :: A(2)=18 :: B(2)=50 :: A(3)=50
120 B(3)=35 :: A(4)=70 :: B(4)=15 :: A(5)=70 :: B(5)=45
130 GOTO 180
140 CALL CLEAR :: INPUT "HOW MANY CHARGES ":N
150 FOR I=1 TO N
160 PRINT "TYPE X AND Y POSITIONS OF CHARGE " :: INPUT A(I),B(I)
170 NEXT I
180 F=SQR(3)/2
190 FOR I=1 TO N :: A(I)=A(I)*F :: NEXT I
200 C(1)=1 :: C(2)=1 :: C(3)=-2 :: C(4)=-1 :: C(5)=1 :: GOTO 250
210 PRINT "TYPE IN THE SIZE OF EACH CHARGE. USE 3 FOR 3Q"
220 FOR I=1 TO N
230 PRINT "CHARGE ";:: I;:: INPUT C(I):: NEXT I
240 F=SQR(3)/2
250 FOR J=1 TO 121 STEP 2
260 X=J*F
270 FOR Y=1 TO 61
280 GOSUB 1000
290 Z=(V+1)*26
300 IF Z<1 THEN B$=" " :: GOTO 330
310 IF Z>52 THEN B$="@" :: GOTO 330
320 IF Z>INT(Z)+.5 THEN B$=" " ELSE B$=A$(Z)
330 PRINT #1:B$;
340 NEXT Y
350 PRINT #1
360 NEXT J
370 CALL KEY(0,N,S):: IF S=0 THEN 380 ELSE 9999
1000 !subroutine to calculate field strength
1010 V=0
1020 FOR I=1 TO N
1030 X1=X-A(I):: Y1=Y-B(I)
1040 R=SQR(X1*X1+Y1*Y1)
1050 IF R>0 THEN V=V+C(I)/R ELSE V=10000 :: RETURN
1060 NEXT I
1070 RETURN
1080 CLOSE #1
9999 END
```



Discussion

- Notice that the conversational portion of the program has been bypassed by the statements at 110-130 and 180-200. This was done so that a test could be run with 5 charges located on the rectangular area.

- Beware! This program is a number-crunching sloth. You must have patience with the slowness of the output. It is quite extraordinary looking when complete.

Suggestions

- Rewrite the program to display other field relationships. To increase the range of intensity from -2 and $+2$ to, say, -10 and $+10$, change line 180 to
  180 F = SQR(3)/10

- To change the characters that represent the various intensities, rewrite the DATA statements in lines 70 and 80 and the loops in 90 and 100. They are presently written to use 52 ASCII characters.

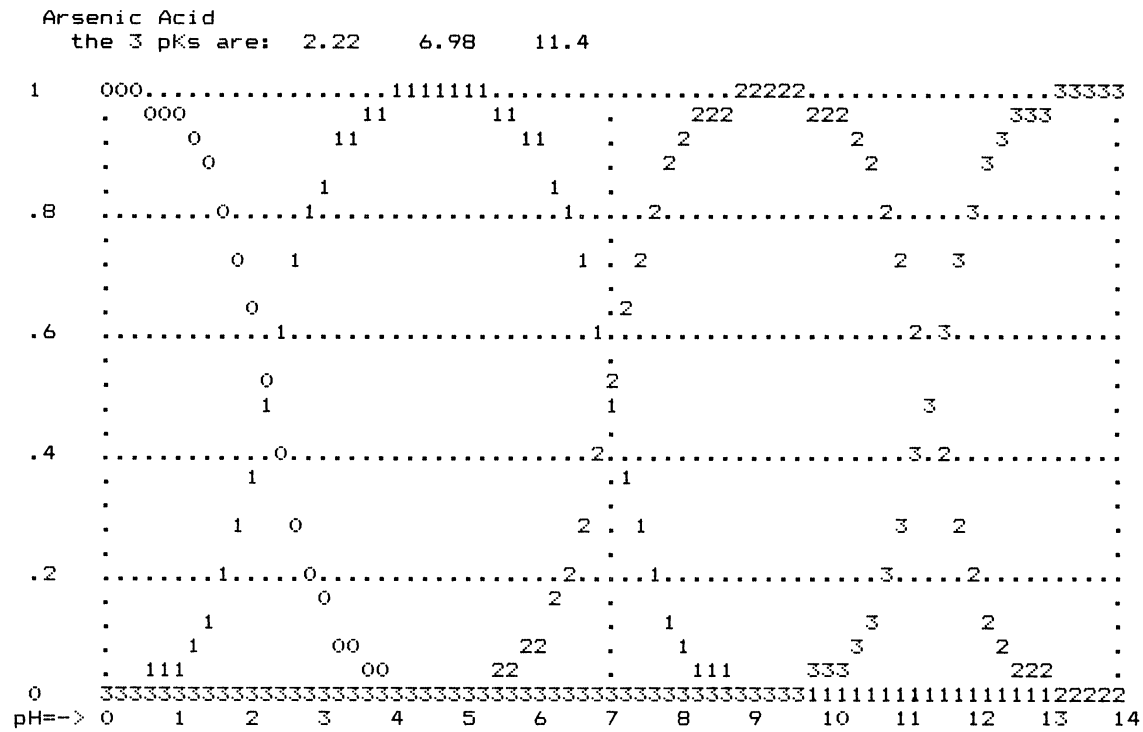
Problem 4.4

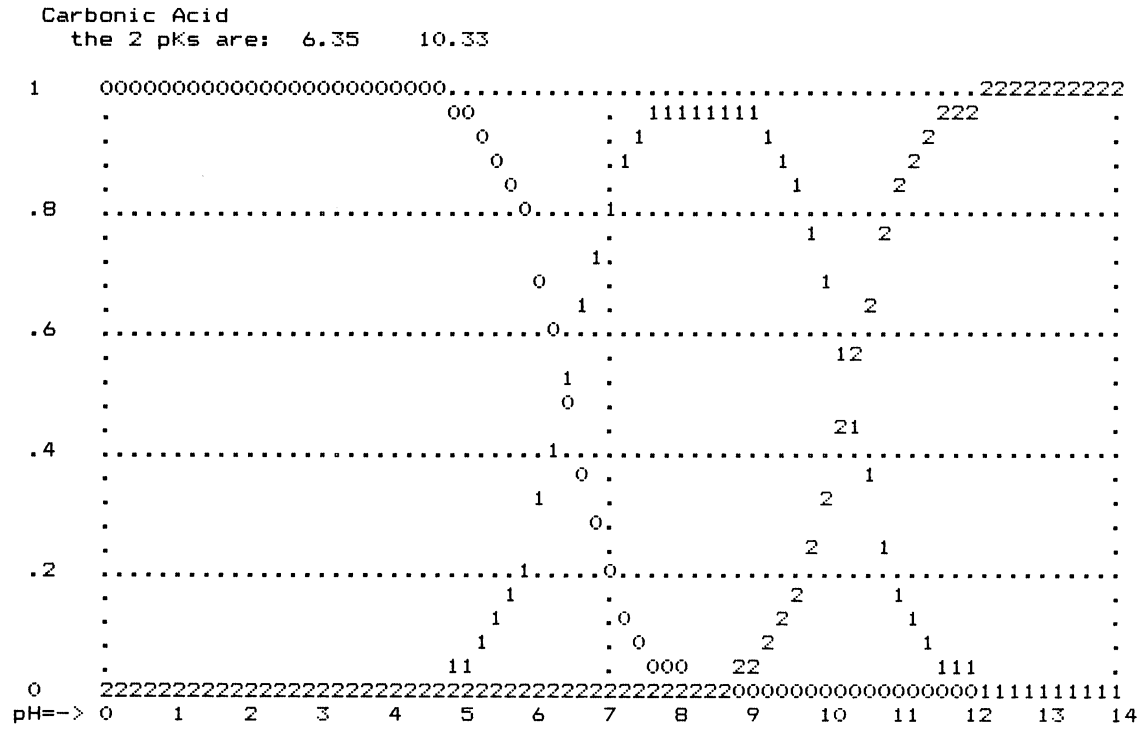
Display the varying concentrations of up to six ionic species that may coexist in a solution of varying acidity. Solving this classic problem in analytic chemistry will provide a chemist with the ability to determine at a glance what the "soup" of acid contains at a particular level of acidity, or pH. Without such a graphical aid, the chemist must rely on rather complex and time-consuming calculations to obtain the same information.

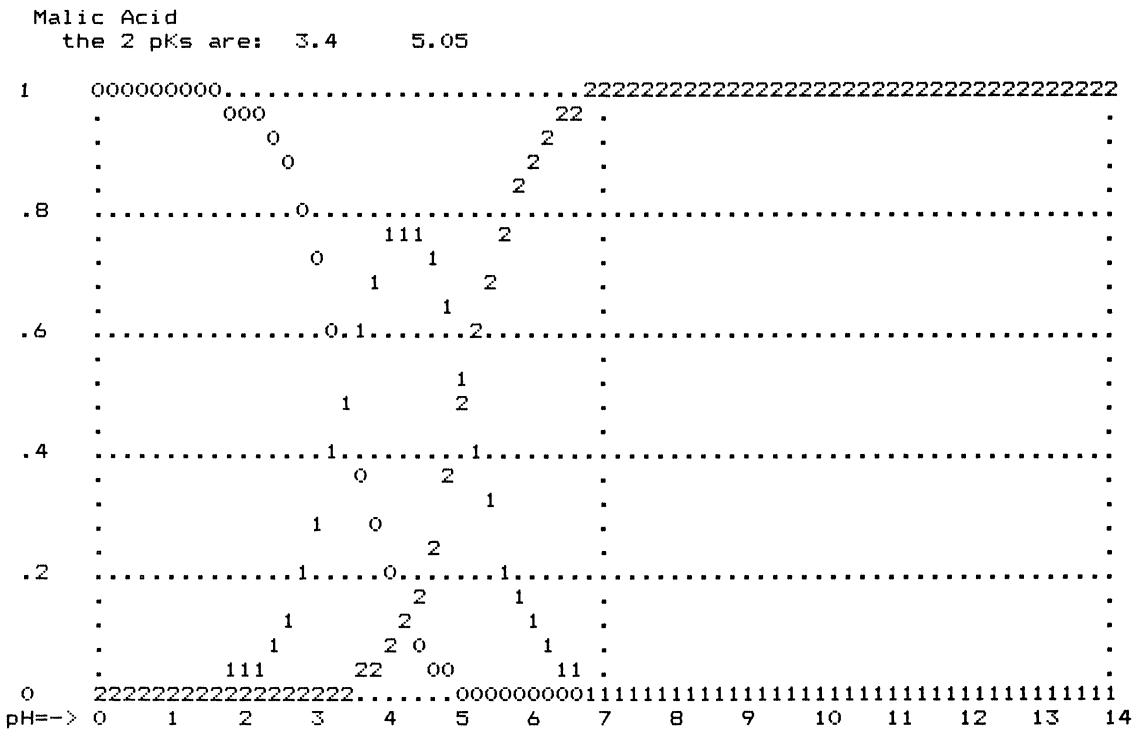
Solution

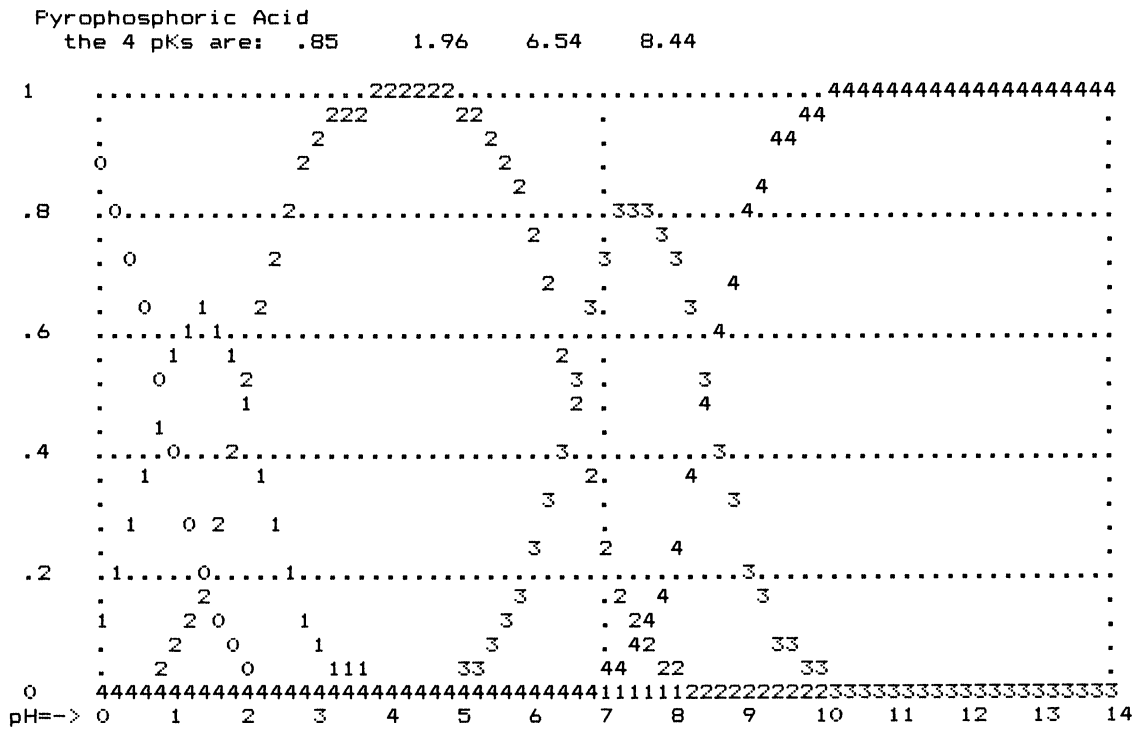
```
10 ! filename:"gr4p4"
20 !  purpose: chemical soup graphing
30 !   author: jpg, jdr & tfz 10/83
40 !
50 DIM G$(26),PK(6),K(6),A(6),Y(6)
60 OPEN #1:"RS232"
70 W=71 :: WI=5 :: !set up width to 71 cols, intervals to 5
80 INPUT "Which included acid from the DATA stmts ":X
90 FOR J=1 TO X :: READ A$,N :: Y(0)=1
100 FOR I=1 TO N :: READ PK(I)
110 ! Get ionization constants, cumulative products
120 K(I)=10^(-PK(I)):: Y(I)=K(I)*Y(I-1)
130 NEXT I
140 NEXT J
150 PRINT #1 :: PRINT #1
160 PRINT #1:TAB(3);A$
170 PRINT #1:TAB(5);"the";N;"pKs are:";
180 FOR I=1 TO N :: PRINT #1:TAB(I*8+12);PK(I);:: NEXT I
190 PRINT #1 :: PRINT #1
200 !  Prepare the background grid
210 !  First, blank out inner strings
220 FOR I=2 TO 25 :: G$(I)=RPT$(" ",W-2):: NEXT I
230 !  Then dot the outer, interval strings
240 FOR I=1 TO 26 STEP 5 :: G$(I)=RPT$(".",W-2):: NEXT I
250 !  Then dot left, right, center columns
260 FOR I=1 TO 26
270 G$(I)="."&SEG$(G$(I),1,34)&"."&SEG$(G$(I),LEN(G$(I))-34,34)&"."
```

```
280 NEXT I
290 !  Go through pH range
300 !  Calculate H-ion concentration
310 FOR PH=0 TO 14 STEP .2 :: PRINT PH;:: H=10^(-PH):: D=0
320 !  Calculate H-ion cululative products, denom.
330 FOR I=0 TO N :: Z(I)=H^(N-I)*Y(I):: D=D+Z(I):: NEXT I
340 !  Calculate Alpha value (activity of subspecies)
350 FOR I=0 TO N :: A(I)=Z(I)/D
360 !  Calculate vertical displacement V (which string)
370 !  and horizontal displacement B
380 V=INT(A(I)*25+1.5):: B=PH*WI+1
390 !  Substitute appropriate digit
400 !  in proper position of proper string
410 G$(V)=SEG$(G$(V),1,B-1)&CHR$(48+I)&SEG$(G$(V),B+1,LEN(G$(V)))
420 NEXT I
430 NEXT PH
440 PRINT
450 T=1.2 :: !  T is vertical concentration marker 0 to 1.0
460 FOR I=26 TO 1 STEP -1
470 IF INT((I-1)/5)*5<>I-1 THEN 510
480 T=T-.2
490 IF T<.2 THEN T=0
500 PRINT #1:T;
510 PRINT #1:TAB(7);G$(I)
520 NEXT I
530 PRINT #1:"pH=->";
540 FOR I=0 TO 14 :: PRINT #1:TAB(I*WI+6);I;:: NEXT I
550 PRINT #1
560 STOP
1000 !  ***** data statements with acid names, pks
1010 DATA "Ethylene Diamine Tetra-acetic Acid (EDTA)"
1020 DATA 4,2.00,2.67,6.16,10.26
1030 DATA "Arsenic Acid",3,2.22,6.98,11.4
1040 DATA "Carbonic Acid",2,6.35,10.33
1050 DATA "Citric Acid",3,3.13,4.76,6.40
1060 DATA "Malic Acid",2,3.40,5.05
1070 DATA "Oxalic Acid",2,1.27,4.27
1080 DATA "Phosphoric Acid",3,2.15,7.20,12.4
1090 DATA "Pyrophosphoric Acid",4,.85,1.96,6.54,8.44
1100 DATA "Salicylic Acid",2,2.97,13.0
1110 DATA "Tartaric Acid",2,3.04,4.37
1120 DATA "DCTA",4,2.40,3.52,6.12,11.7
1130 CLOSE #1
9999 END
```


Arsenic Acid
the 3 pKs are: 2.22 6.98 11.4
pH=-> 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14


Carbonic Acid
the 2 pKs are: 6.35 10.33
pH=-> 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14


Malic Acid
the 2 pKs are: 3.4 5.05
pH=-> 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14


Pyrophosphoric Acid
the 4 pKs are: .85 1.96 6.54 8.44
pH=-> 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14

Discussion

- A wide variety of disciplines are plagued with experimental conditions such as these. Interfering signals in electronics, competing species in genetics, varying levels of soil conditioners, fertilizers and hybrids in agriculture—all lend themselves to this graphing technique. The calculations for each application are different, but the terminal result is the same: The experimenter wants a graph of competing, interfering, or mutually affecting species.

Suggestions

- RUN this program with various acids to get a feel for the way pKs affect competing species. Here is a list of acids and their pKs for you to try.

Name	pK1	pK2	pK3	pK4
Arsenic acid	2.22	6.98	11.4	
Carbonic acid	6.35	10.33		
Citric acid	3.13	4.76	6.40	
Malic acid	3.40	5.05		
Oxalic acid	1.27	4.27		
Phosphoric acid	2.15	7.20	12.4	
Pyrophosphoric acid	0.85	1.96	6.54	8.44
Salicylic acid	2.97	13.0		
Tartaric acid	3.04	4.37		
EDTA	2.00	2.67	6.16	10.26
DCTA	2.40	3.52	6.12	11.7

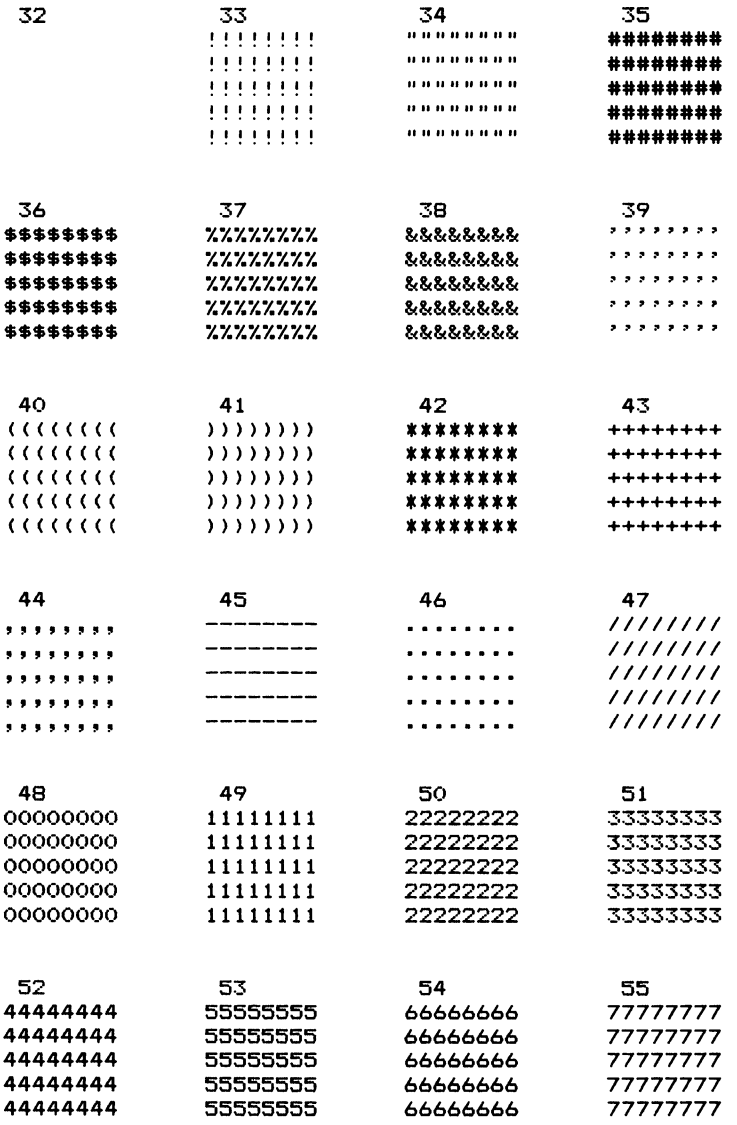
- Add a table-printing feature to this program that prints out the concentrations of all species present at each pH.

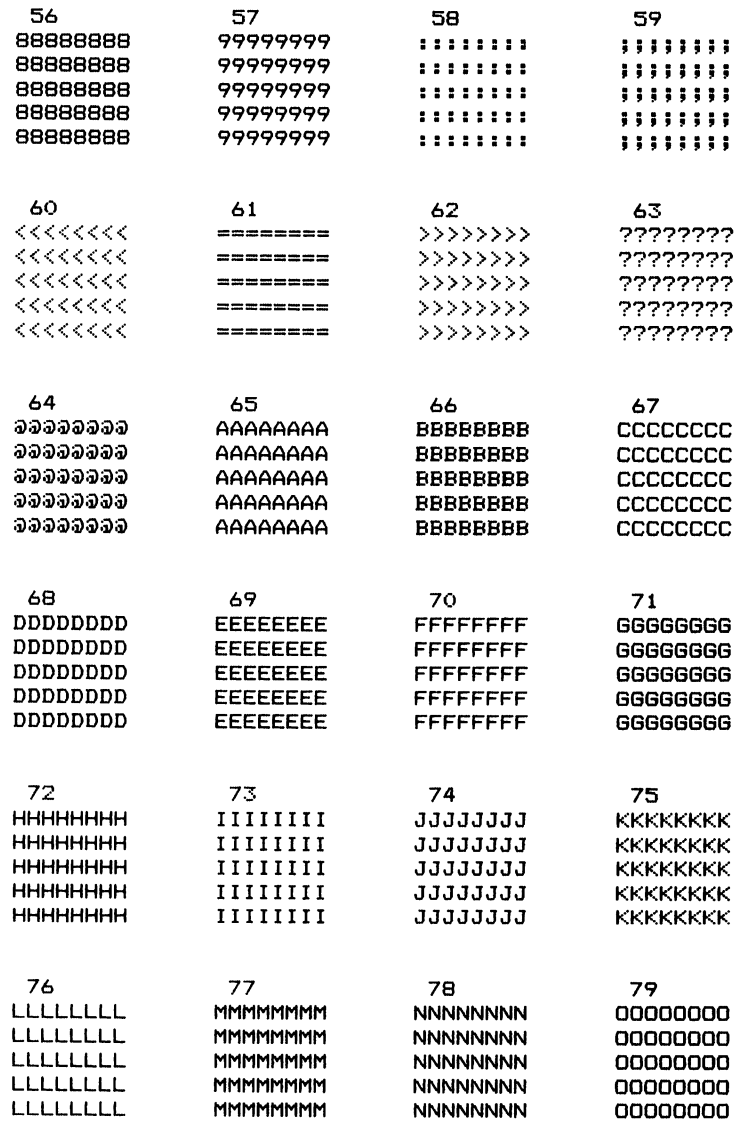
Problem 4.5

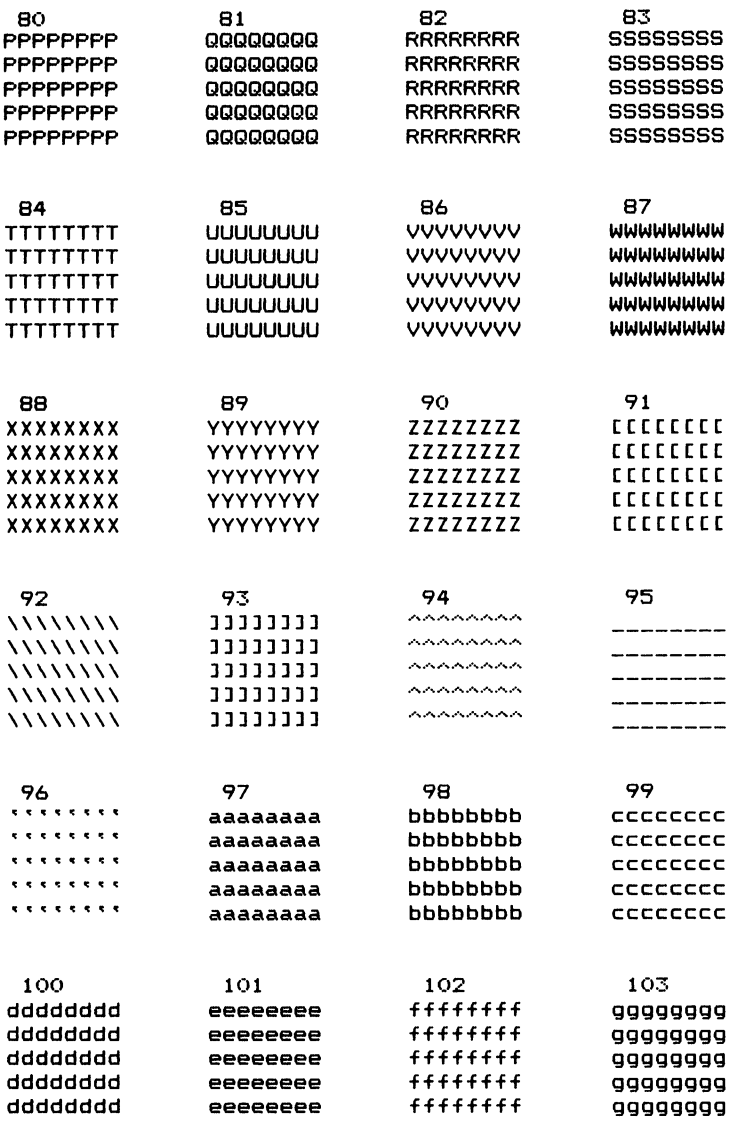
Produce a chart on the printer that displays blocks of all printable characters so that an enterprising programmer can scan it to pick out likely candidates for future density gradient graphs. Each block is to be 5 characters tall and 8 characters wide, and separated from its neighboring blocks by 8 spaces. Above each block is to be printed the ASCII code value which that character represents.

Solution

```
10 ! filename:"gr4p5"
20 !  purpose: produce a chart to show character densities
30 !    author: jpg, jdr & tfz 10/83
40 !
45 OPEN #1:"RS232"
50 I=32
60 FOR L=1 TO 4
70 FOR K=1 TO 6
80 PRINT #1:I,I+1,I+2,I+3
90 FOR J=1 TO 5
100 PRINT #1:RPT$(CHR$(I),8),:: PRINT #1:RPT$(CHR$(I+1),8),
110 PRINT #1:RPT$(CHR$(I+2),8),:: PRINT #1:RPT$(CHR$(I+3),8)
120 NEXT J
130 PRINT #1 :: PRINT #1
140 I=I+4
150 NEXT K
160 FOR K=1 TO 8
170 PRINT #1
180 NEXT K
190 NEXT L
200 CLOSE #1
999 END
```


32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55


56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79


80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103

```
 104       105       106       107
hhhhhhhh  iiiiiiii  jjjjjjjj  kkkkkkkk
hhhhhhhh  iiiiiiii  jjjjjjjj  kkkkkkkk
hhhhhhhh  iiiiiiii  jjjjjjjj  kkkkkkkk
hhhhhhhh  iiiiiiii  jjjjjjjj  kkkkkkkk
hhhhhhhh  iiiiiiii  jjjjjjjj  kkkkkkkk

 108       109       110       111
llllllll  mmmmmmmm  nnnnnnnn  oooooooo
llllllll  mmmmmmmm  nnnnnnnn  oooooooo
llllllll  mmmmmmmm  nnnnnnnn  oooooooo
llllllll  mmmmmmmm  nnnnnnnn  oooooooo
llllllll  mmmmmmmm  nnnnnnnn  oooooooo

 112       113       114       115
pppppppp  qqqqqqqq  rrrrrrrr  ssssssss
pppppppp  qqqqqqqq  rrrrrrrr  ssssssss
pppppppp  qqqqqqqq  rrrrrrrr  ssssssss
pppppppp  qqqqqqqq  rrrrrrrr  ssssssss
pppppppp  qqqqqqqq  rrrrrrrr  ssssssss

 116       117       118       119
tttttttt  uuuuuuuu  vvvvvvvv  wwwwwwww
tttttttt  uuuuuuuu  vvvvvvvv  wwwwwwww
tttttttt  uuuuuuuu  vvvvvvvv  wwwwwwww
tttttttt  uuuuuuuu  vvvvvvvv  wwwwwwww
tttttttt  uuuuuuuu  vvvvvvvv  wwwwwwww

 120       121       122       123
xxxxxxxx  yyyyyyyy  zzzzzzzz  {{{{{{{{
xxxxxxxx  yyyyyyyy  zzzzzzzz  {{{{{{{{
xxxxxxxx  yyyyyyyy  zzzzzzzz  {{{{{{{{
xxxxxxxx  yyyyyyyy  zzzzzzzz  {{{{{{{{
xxxxxxxx  yyyyyyyy  zzzzzzzz  {{{{{{{{

 124       125       126       127
||||||||  }}}}}}}}  ~~~~~~~~
||||||||  }}}}}}}}  ~~~~~~~~
||||||||  }}}}}}}}  ~~~~~~~~
||||||||  }}}}}}}}  ~~~~~~~~
||||||||  }}}}}}}}  ~~~~~~~~
```

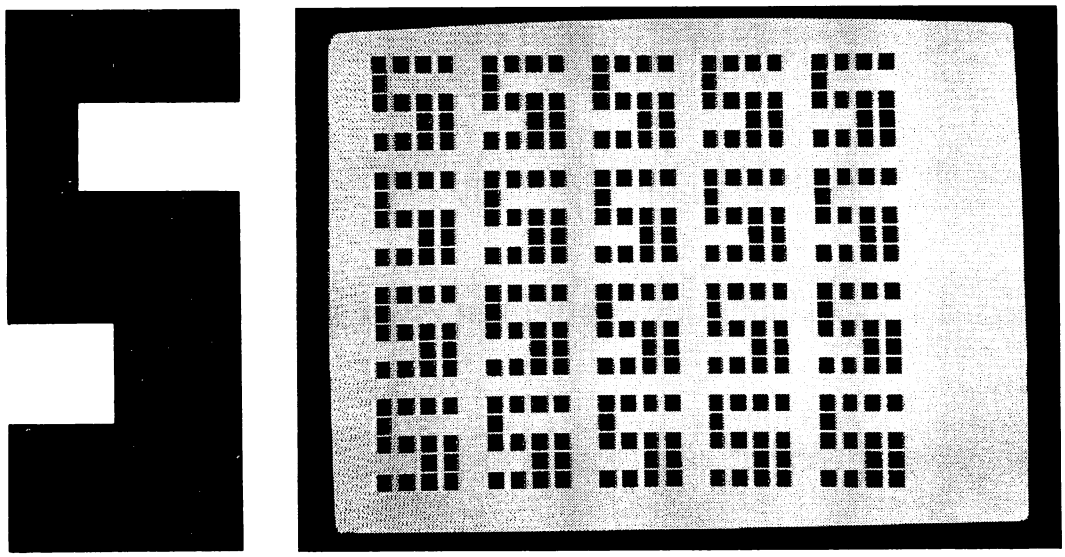


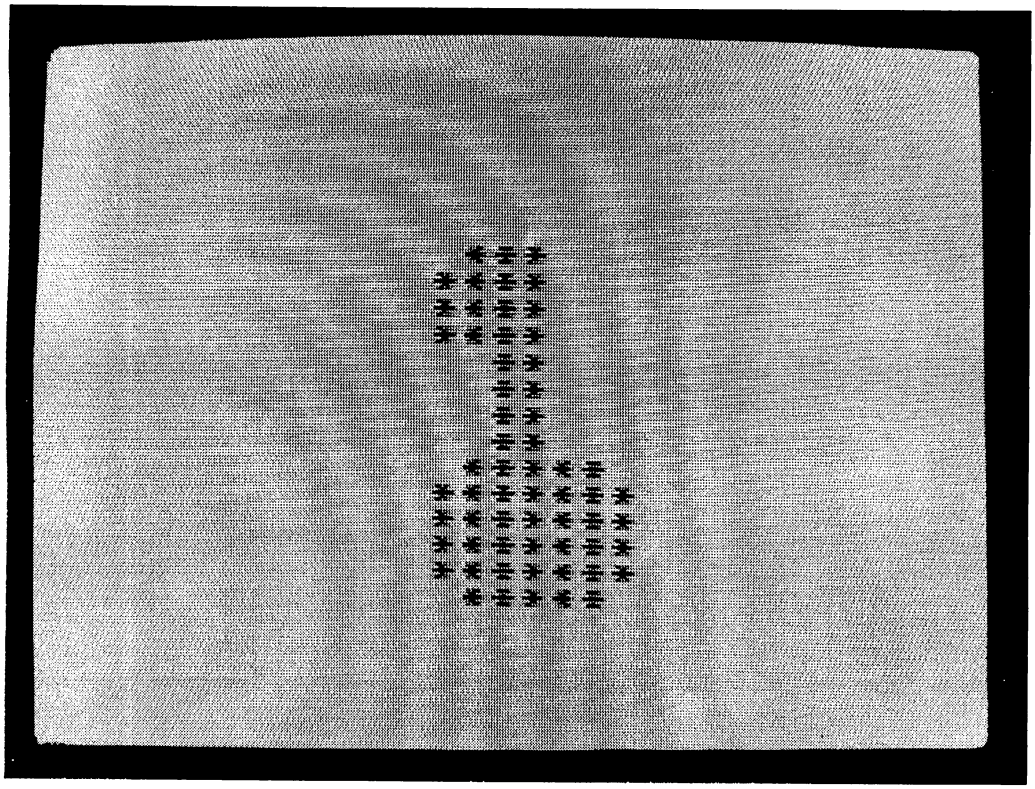
Table-Driven Pictures

Many graphic designs can be contained within a program as DATA statements whose values represent an encoded version of the picture or design. The program READs the DATA values and performs the decoding necessary to reconstruct the graphic design.

Problem 5.1 Draw an expanded digit 1 on the screen.

Solution Store the sketch of the digit into a 2-dimensional table in which the first column represents the starting position in a line of output of a string of asterisks. The second column of the table specifies the length of the string.

```
10 ! filename:"gr5p1"
20 !  purpose: draw expanded digit 1, coded by position & length
30 !   author: jpg, jdr & tfz 10/83
40 !
50 CALL CLEAR
60 DIM D(14,2)
70 FOR I=1 TO 14
80 READ D(I,1),D(I,2)
90 NEXT I
100 DATA 13,3,12,4,12,4,12,4,14,2,14,2,14,2,14,2
110 DATA 13,5,12,7,12,7,12,7,12,7,13,5
120 FOR I=1 TO 14
130 PRINT TAB(D(I,1));RPT$("*",D(I,2))
140 NEXT I
999 END
```



Discussion

- The graph of a single digit, as in this case, suggests the many such designs that could be produced this way.
- You can expand the technique that this example shows by incorporating a repetition factor, wherein there are three values associated with a PRINT statement:
  (1) number of PRINT statements of this format,
  (2) starting position for the string,
  (3) length of the string.
- In order to fully generalize this technique, two more pieces of information could be added to the encoding information:
  (1) indicator of whether this PRINT line is finished or is to be continued,
  (2) what character is to be used in the string.

Suggestions

- Modify the program by implementing the ideas in the discussion points above. Cause the program to draw an enlarged digit 0.

- Modify the program so that the user can select the position where the digit is to be placed on the screen.
- Create graphic symbols and make up DATA statements that could be used by your program for display on the screen.

Problem 5.2

Draw an expanded digit 1 on the screen.

Solution

Encode the sketch of the digit into a series of binary numbers, with zeros and ones representing the absence or presence of a character. For example, each line of a sketch of the digit 1

```
  xxx
 xxxx
 xxxx
 xxxx
   xx
   xx
   xx
   xx
  xxxxxx
 xxxxxxxx
 xxxxxxxx
 xxxxxxxx
 xxxxxxxx
  xxxxxx
```

can be encoded into binary by using a binary digit 1 to represent an x and a binary digit 0 to represent a blank. The result would be the 14 8-bit binary numbers

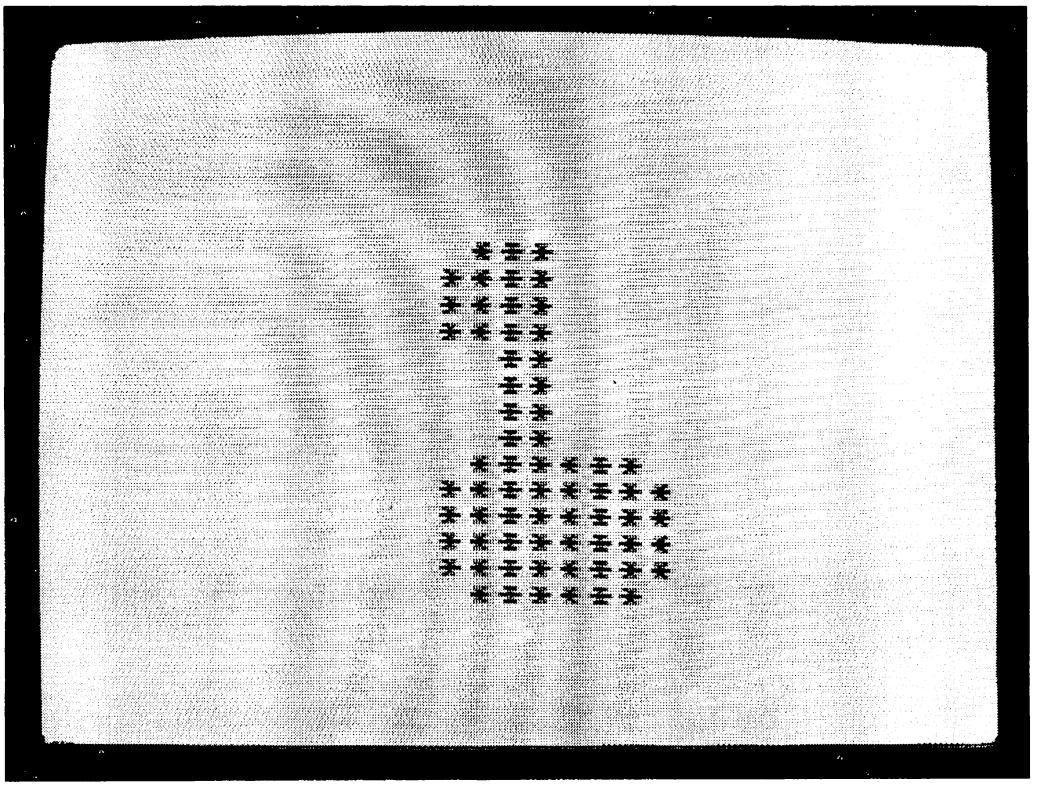
```
01110000
11110000
11110000
11110000
00110000
00110000
00110000
00110000
01111110
11111111
11111111
11111111
11111111
01111110
```

Convert each of the binary numbers to its decimal equivalent. Remember that the binary number system uses each place to represent a power of two. The rightmost position corresponding to 2 to the zero power or 1, the next being 2 to the first power or 2, the next being 2 to the second power or 4, and so on. Thus the binary numbers above equal the decimal numbers

112
240
240
240
48
48
48
48
123
255
255
255
255
123

All the program needs to do then is decode the decimal numbers (convert them back to binary) and cause an asterisk to correspond to a 1 and a blank to correspond to a 0.

```
10 ! filename:"gr5p2"
20 !  purpose: draw expanded digit 1, coded in binary
30 !   author: jpg, jdr & tfz 10/83
40 !
50 CALL CLEAR
60 DIM D(14),X$(14)
70 FOR I=1 TO 14
80 READ D(I)
90 NEXT I
100 DATA 112,240,240,240,48,48,48,48
110 DATA 126,255,255,255,255,126
120 FOR I=1 TO 14
130 C=D(I)
140 X$(I)=""
150 FOR J=1 TO 8
160 Q=INT(C/2)
170 R=C-2*Q
180 IF R=1 THEN X$(I)="*"&X$(I)ELSE X$(I)=" "&X$(I)
190 C=Q
200 NEXT J
210 NEXT I
220 FOR I=1 TO 14
230 PRINT TAB(12);X$(I)
240 NEXT I
999 END
```



Discussion

- Because of the internal representation of integers in the TI–99/4A, we are able to code each line of the figure in binary if its width is less than sixteen characters wide. Of course, wider pictures could use two or more numbers to represent each line of the picture.

- Many other kinds of designs can be represented easily this way. For example, an alphabet or other set of graphic symbols could be coded and thus displayed in a manner similar to that used in this program.

Suggestions

- Alter the program so that the one becomes twice as fat when displayed, but don't change any of the encoded data values.

- Alter the program so that the user can specify the position of the number on the screen. Also, let the user specify the character used to sketch the figure.

- Design a corporate logo or other simple figure and place the encoded information in DATA statements and alter the program to output it.

Problem 5.3 Draw the silhouette of a witch.

Solution

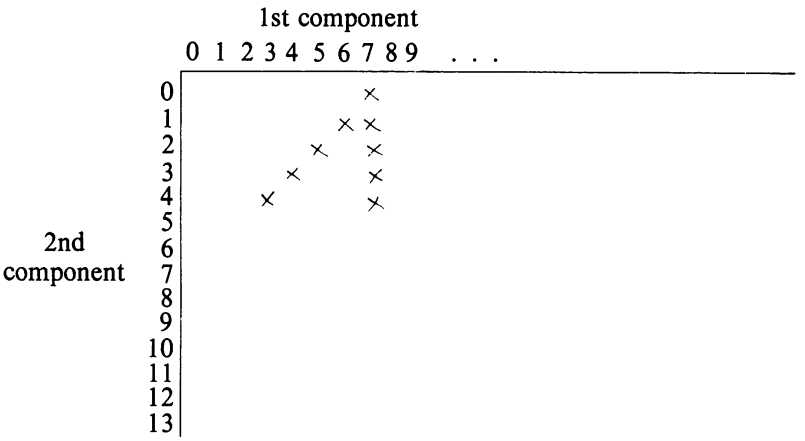
```
10 ! filename:"gr5p3"
20 !  purpose: draw a silhouette of a witch
30 !   author: jpg, jdr, spg& tfz 10/83
40 !
45 OPEN #1:"RS232"
50 READ A$ :: ! get next piece of data
60 IF A$="END" THEN PRINT #1 :: GOTO 250
70 ! check for a new line
80 IF A$="B" THEN PRINT #1 :: PRINT #1:TAB(1);:: GOTO 50
90 ! check for the code for spaces (s##)
100 IF SEG$(A$,1,1)="S" THEN PRINT #1:RPT$(" ",VAL(SEG$(A$,2,2)));:: GOTO 50
110 PRINT #1:RPT$("%",VAL(A$));
120 GOTO 50
130 DATA B,S44,2,B,S43,4,B,S43,5,B,S44,6,B,S45,5,B,S45
140 DATA 6,B,S46,6,B,S46,7,S7,2,B,S45,9,S5,3,B,S45,16,B,S45,15
150 DATA B,S45,14,B,S42,17,B,S40,20,B,S40,21,B,S43,16,B,S34,26,B,S31,28,B
160 DATA S28,28,B,S26,30,B,S24,32,B
170 DATA S22,34,B,S20,36,B,S18,39,B,S16,43,B,S15,46,B,S14
180 DATA 54,B,S10,62,S3,3,B,S5,74,B,79,B,S1,77,B,S2,23,S3,45,B,S8,14
190 DATA S6,44,B,S28,36,S3,4,B,S28,37,B,S28,36,B,S29,34,B,S30,32,B,S30,32
200 DATA B,S32,30,B,S37,24,B,S29,2,S2,27,B,S25,34,B
210 DATA S14,5,S1,12,S2,24,B
220 DATA S8,23,S2,24,S9,3,B,28,S3,26,S3,10,B,25,S4,40,B,S1
230 DATA 21,S4,40,B,S3,15,S4,39,B,S3,14,S4,24,B,S4,9,S4,23,B,S6,5,S3
240 DATA 23,B,S6,3,S3,22,B,S6,2,S4,16,B,S14,8
250 DATA END
260 CLOSE #1
270 END
```

Discussion

- This program is a variation on many of the ideas presented in the previous programs. The DATA statements can be read as follows:
  B (begin line)
  S44 (skip 44 spaces)
  2 (mark 2 spaces with characters)
  .
  .
  .

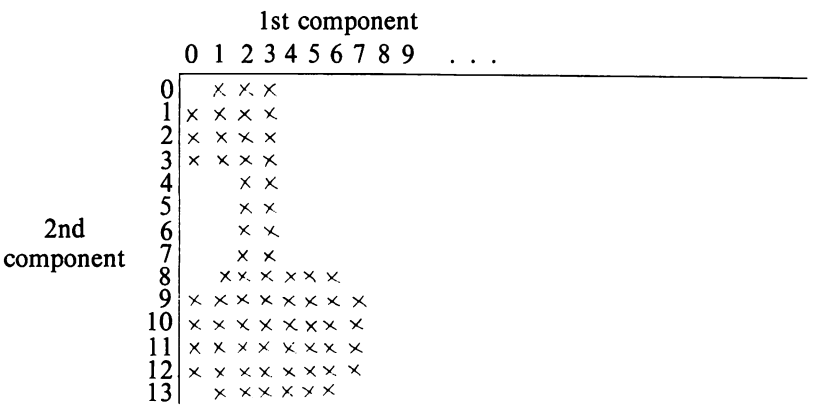
As we near completion of this chapter and the discussion of line printer graphics in general, we will introduce another method for encoding a picture description that will provide a generalized program upon which many other programs can be based to generate graphic designs of all kinds.

The method used to code a picture or design is based on the fact that all pictures produced on a video screen or line printer are formed from single characters or straight lines composed of characters. If a grid is introduced over the picture like that shown below, then the picture or design can be reduced to a set of pairs of end points of a bunch of straight lines.



As an example, the figure in the grid is composed of two straight lines. The first has end points (3,4) and (7,0). The second has end points (7,0) and (7,4). If we were to draw the straight lines connecting these end points and fill in the middle points, the figure shown in the drawing would be created. That is the basis of this method: Reduce the drawing to end points of a straight line. Then plot the endpoints and all points on the line in between.

The expanded digit 1 of programs GR5P1 and GR5P2 could be coded as:



(0,1) to (0,3)
(1,0) to (1,3)
(2,0) to (2,7)
(3,0) to (3,7)
(1,8) to (6,8)
(0,9) to (7,9)
(0,10) to (7,10)
(0,11) to (7,11)
(0,12) to (7,12)
(1,13) to (6,13)

A further compression of this encoding scheme would be to combine the two numbers of each end point into a single number without loss of information. Since the second number will be at most 2 digits, the end point (7,10) could be expressed as the single number 710 with the understanding that the rightmost two digits of such a number represent the vertical grid position or second component. The horizontal grid position or first component is the part of the number left when the rightmost two digits are removed. With such a compression scheme, the expanded digit 1 could be encoded as the series of numbers:

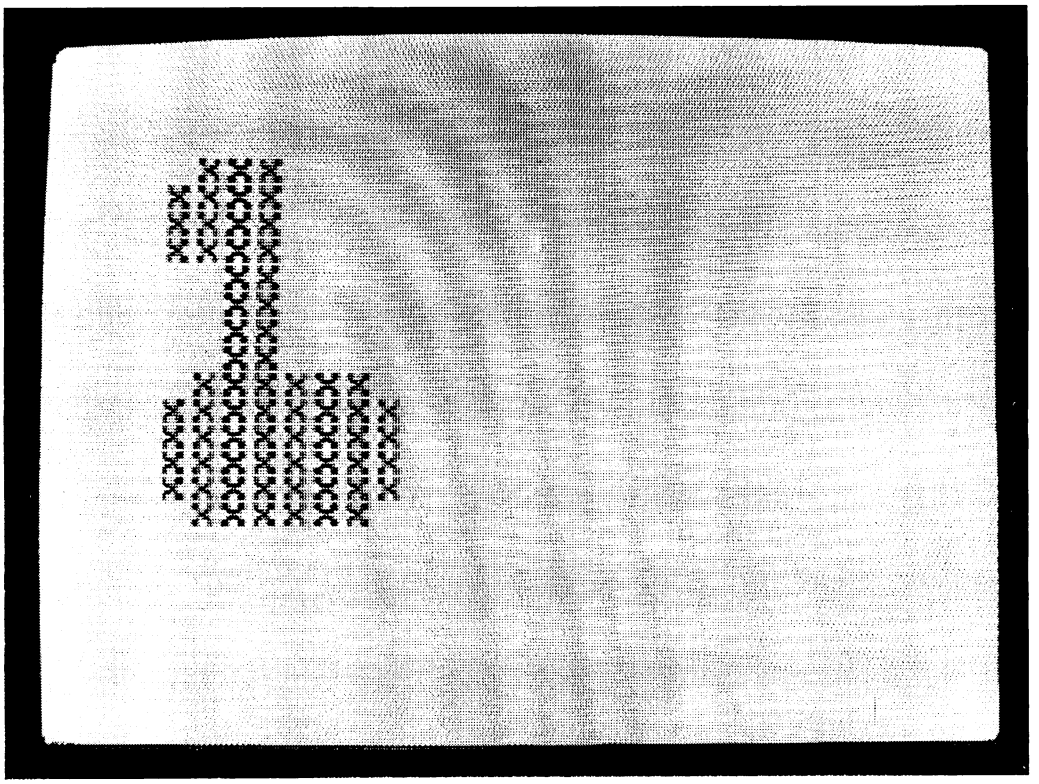
1,3,100,103,200,207,300,307,108,608,9,709,
10,710,11,711,12,712,113,613

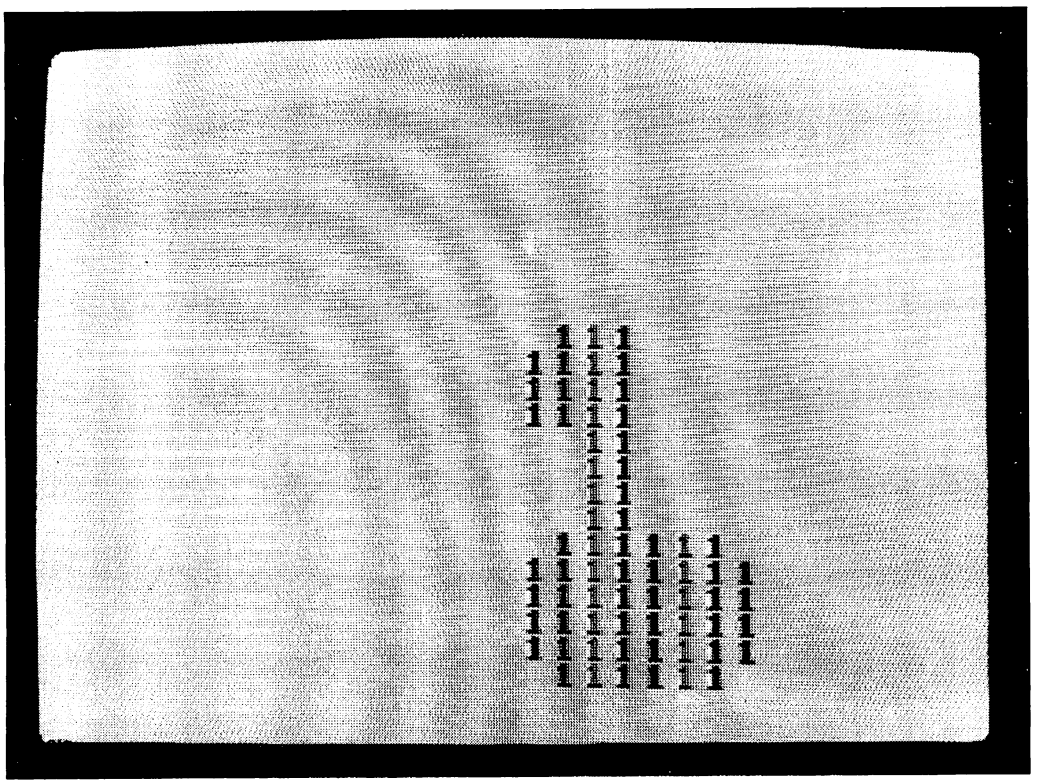
Notice that each pair of numbers are the end points of a line to be drawn. Also notice that the first two numbers (1 and 3) would be more recognizable as 001 and 003, but the leading zeros are redundant when we normally write numbers and were left off with no loss of information.

Problem 5.4 Draw an expanded digit 1 on the video screen using the straight line coding method.

Solution

```
10 ! filename:"gr5p4"
20 !  purpose: draw expanded digit 1, coded as straight lines
30 !   author: jpg, jdr & tfz 10/83
40 !
50 CALL CLEAR
60 INPUT "      input x,y position of picture on screen":XINC,YINC
70 INPUT "      input characters to be used in drawing":CHAR$
80 CHAR$=SEG$(CHAR$,1,5)
90 GOSUB 1000 :: ! draw the picture
100 CALL KEY(0,N,S):: IF S=0 THEN 100 ELSE 9999
1000 ! subroutine to draw picture on screen
1010 CALL CLEAR
1020 READ N :: !number of straight lines to draw in picture
1030 FOR L=1 TO N
1040 READ E,C,D
1050 X1=INT(C/100):: Y1=C-100*X1
1060 X2=INT(D/100):: Y2=D-100*X2
1070 X1=X1+XINC :: Y1=Y1+YINC :: !relocate coordinates
1080 X2=X2+XINC :: Y2=Y2+YINC
1090 IF X1>27 OR X2>27 OR Y1>24 OR Y2>24 THEN PRINT "picture will not fit, canno
t continue" :: RETURN
1100 IF X1=X2 THEN 1260
1110 IF Y1=Y2 THEN 1300
1120 M=(Y2-Y1)/(X2-X1):: !calculate slope of line
1130 B=Y-M*X1 :: !calculate y-intercept
1140 IF M>0 THEN IF M<=1 THEN 1210 ELSE 1160 ELSE IF M>=-1 THEN 1210 ELSE 1150
1150 T=Y1 :: Y1=Y2 :: Y2=T :: !slope between -1 and -infinity
1160 FOR J=Y1 TO Y2 :: !slope between 1 and infinity
1170 I=INT((J-B)/M+.5)
1180 DISPLAY AT(J,I):SEG$(CHAR$,E+1,1);
1190 NEXT J
1200 GOTO 1330
1210 FOR I=X1 TO X2 :: !slope between -1 and 1
1220 J=INT(M*I+B+.5)
1230 DISPLAY AT(J,I):SEG$(CHAR$,E+1,1);
1240 NEXT I
1250 GOTO 1330
1260 FOR J=Y1 TO Y2 :: !no slope, vertical line
1270 DISPLAY AT(J,X1):SEG$(CHAR$,E+1,1);
1280 NEXT J
1290 GOTO 1330
1300 FOR I=X1 TO X2 :: !zero slope, horizontal line
1310 DISPLAY AT(Y1,I):SEG$(CHAR$,E+1,1);
1320 NEXT I
1330 NEXT L
1340 RETURN
2000 DATA 10,0,1,3,0,100,103,0,200,207,0,300,307,0,108,608
2010 DATA 0,9,709,0,10,710,0,11,711,0,12,712,0,113,613
9999 END
```





Discussion

- The number zero has been inserted in front of each pair of numbers that represents the end points of a straight line. Its purpose is to specify which character is to be used when drawing the straight line. The effect here is minimal since the same character is used to draw the entire picture. But as will be shown in the next program, this number will not always be zero and will provide for flexibility in the characters used to draw the picture.

- The picture can be placed anywhere on the screen because when the picture was coded we located it in the upper left-hand corner. The program reads in XINC and YINC values which are added to each point to relocate it on the screen.

- It is necessary that the two numbers representing the end points of a line are in numeric order; that is, the first is less than or equal to the second. For example, 3 then 307 is okay, but 307 then 3 is not acceptable. Of course, the program could be modified to place them in order.

Problem 5.5

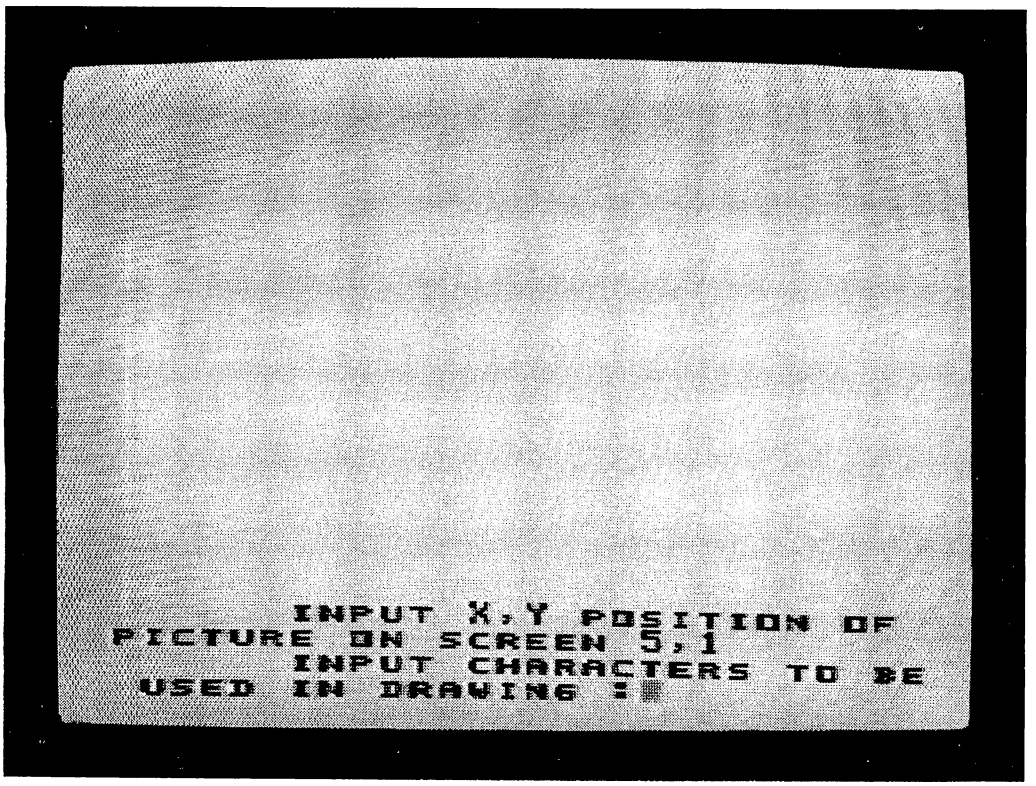
Draw the logo for Bentley College with an optional user's initials to be embedded within it.

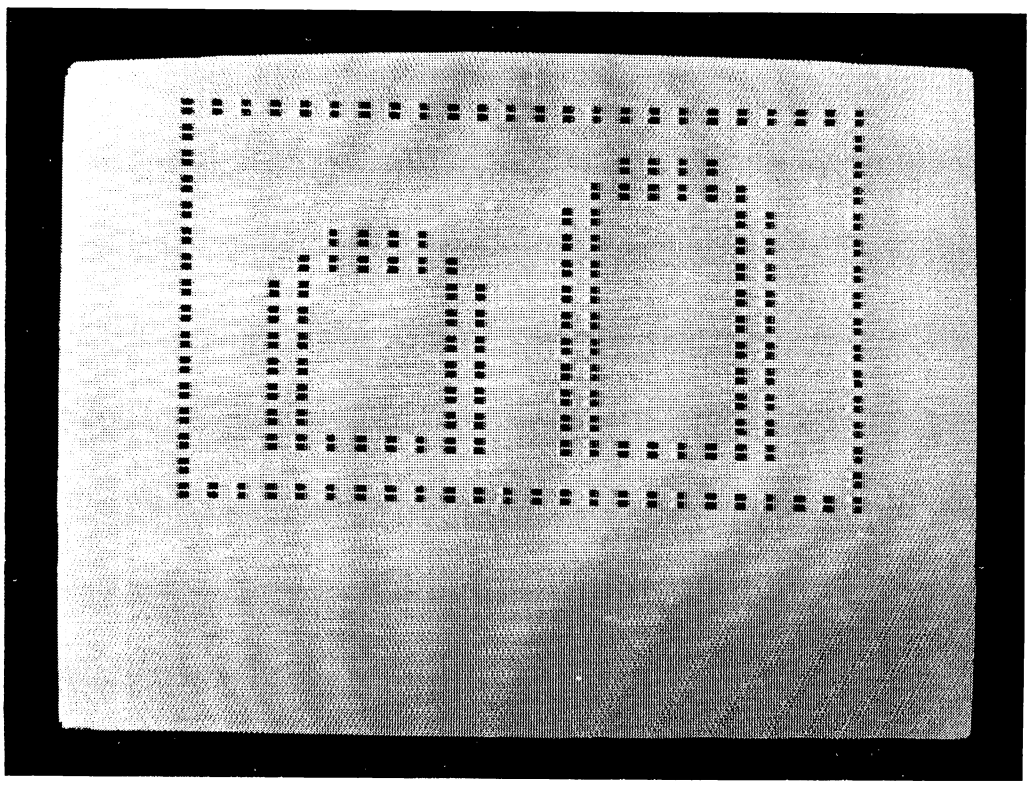
Solution

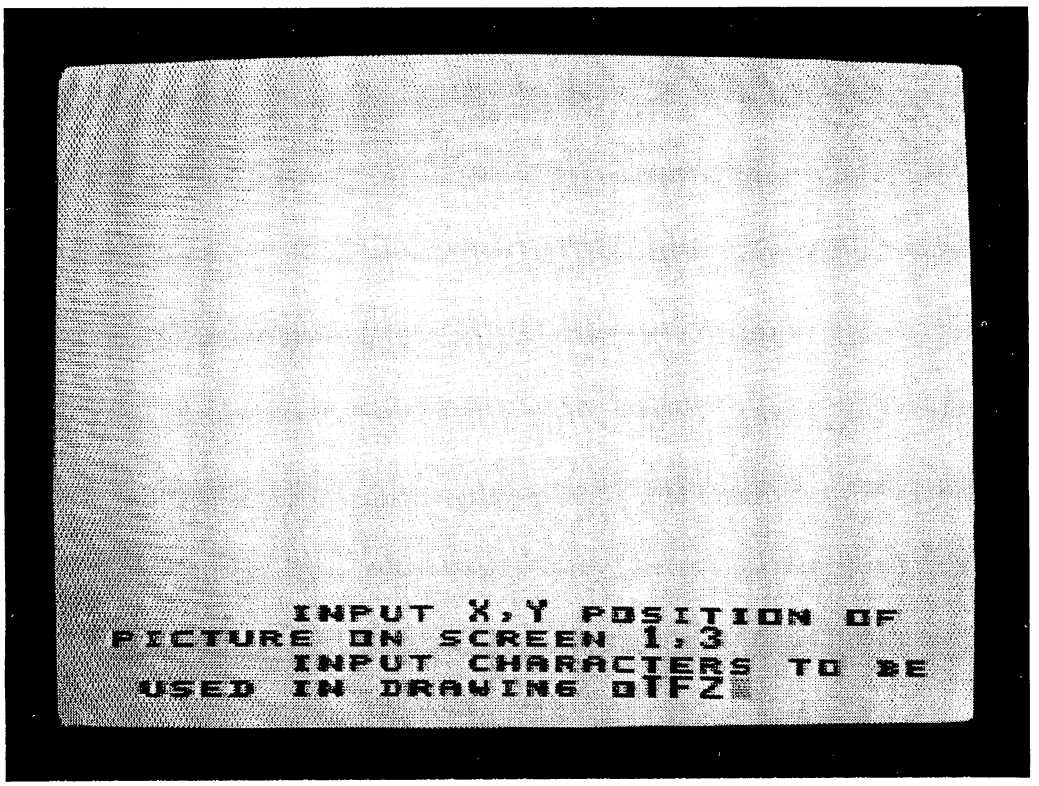
The previous program, with changes only to the DATA statements, will be used to draw the logo.

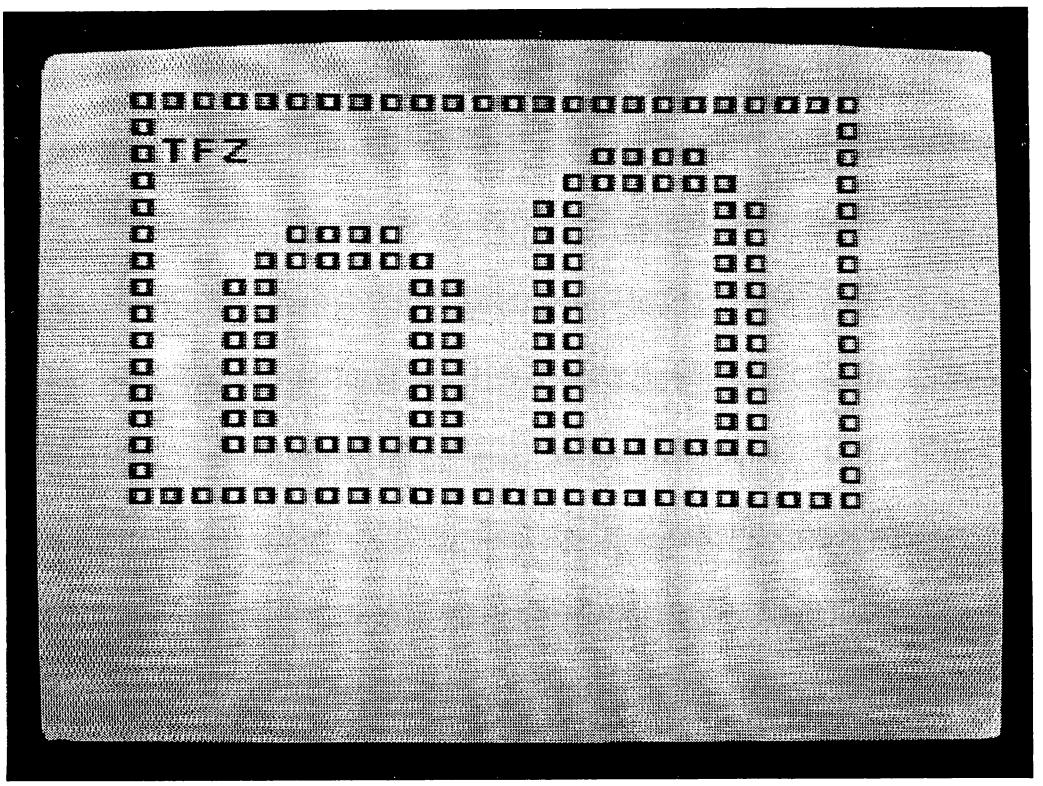
```
10 ! filename:"gr5p5"
20 !  purpose: draw Bentley College logo with user's initials
30 !   author: jpg, jdr & tfz 10/83
40 !
50 CALL CLEAR
60 INPUT "      input X,Y position of picture on screen ":XINC,YINC
70 INPUT "      input characters to be used in drawing ":CHAR$
80 CHAR$=SEG$(CHAR$,1,20)
90 GOSUB 1000 :: !draw the picture
100 CALL KEY(0,N,S):: IF S=0 THEN 100 ELSE 9999
1000 !  Subroutine to draw picture on screen
1010 CALL CLEAR
1020 READ N :: !number of straight lines to draw in picture
1030 FOR L=1 TO N
1040 READ E,C,D
1050 X1=INT(C/100):: Y1=C-100*X1
1060 X2=INT(D/100):: Y2=D-100*X2
1070 X1=X1+XINC :: Y1=Y1+YINC :: !relocate coordinates
1080 X2=X2+XINC :: Y2=Y2+YINC
1090 IF X1/2>28 OR X2/2>28 OR Y1/2>24 OR Y2/2>24 THEN PRINT "picture will not fi
t,       cannot continue" :: RETURN
```

```
1100 IF X1=X2 THEN 1260
1110 IF Y1=Y2 THEN 1300
1120 M=(Y2-Y1)/X2-X1):: !calculate slope of line
1130 B=Y1-M*X1 :: !calculate y-intercept
1140 IF M>0 THEN IF M<=1 THEN 1210 ELSE 1160 ELSE IF M>=-1 THEN 1210 ELSE 1150
1150 T=Y1 :: Y1=Y2 :: Y2=T :: !slope between -1 and -infinity
1160 FOR J=Y1 TO Y2 :: !slope between 1 and infinity
1170 I=INT((J-B)/M+.5)
1180 DISPLAY AT(J,I):SEG$(CHAR$,E+1,1);
1190 NEXT J
1200 GOTO 1330
1210 FOR I=X1 TO X2 :: !slope between -1 and 1
1220 J=INT(M*I+B+.5)
1230 DISPLAY AT(J,I):SEG$(CHAR$,E+1,1);
1240 NEXT I
1250 GOTO 1330
1260 FOR J=Y1 TO Y2 :: !no slope, vertical line
1270 DISPLAY AT(J,X1/2):SEG$(CHAR$,E+1,1);
1280 NEXT J
1290 GOTO 1330
1300 FOR I=X1 TO X2 :: !zero slope, horizontal line
1310 DISPLAY AT(Y1,I/2):SEG$(CHAR$,E+1,1);
1320 NEXT I
1330 NEXT L
1340 RETURN
2000 DATA 31,0,0,4700,0,1,14,0,101,114,0,4601,4614,0,4701,4714
2010 DATA 0,15,4715,0,1005,1705,0,806,1906,0,607,612,0,707,712
2020 DATA 0,807,812,0,907,912,0,1807,1812,0,1907,1912
2030 DATA 0,2007,2012,0,2107,2112,0,613,2113,0,3002,3702
2040 DATA 0,2803,3903,0,2604,2612,0,2704,2712,0,2804,2812
2050 DATA 0,2904,2912,0,3804,3812,0,3904,3912,0,4004,4012
2060 DATA 0,4104,4112,0,2613,4113,1,302,302,2,602,602,3,802,802
9999 END
```


INPUT X,Y POSITION OF
PICTURE ON SCREEN 5,1
INPUT CHARACTERS TO BE
USED IN DRAWING :




INPUT X,Y POSITION OF
PICTURE ON SCREEN 1,3
INPUT CHARACTERS TO BE
USED IN DRAWING TFZ


TFZ

Discussion

- The digit zero isn't in front of each pair of end points. The last 3 pairs have the digits 1, 2, and 3 in front. This means that four characters will be used in drawing the picture. The same character will be used for drawing most of the picture, but the last three lines will be drawn using different characters.

- The last three pairs of end points are strange. Both end points of the line are the same. This means that the line is really just a point, which is a special case of the more general line. The line drawing portion of the subroutine will handle this properly.

- Notice that the input to this program is the character to be used to draw the Bentley College logo followed by the user's initials. If the users initials are missing, then the last three lines are plotted with blanks.

Suggestions

- Investigate the broad generality of this program by using it to draw a number of different pictures in various positions on the screen. The only changes to the program will be in the DATA statements.

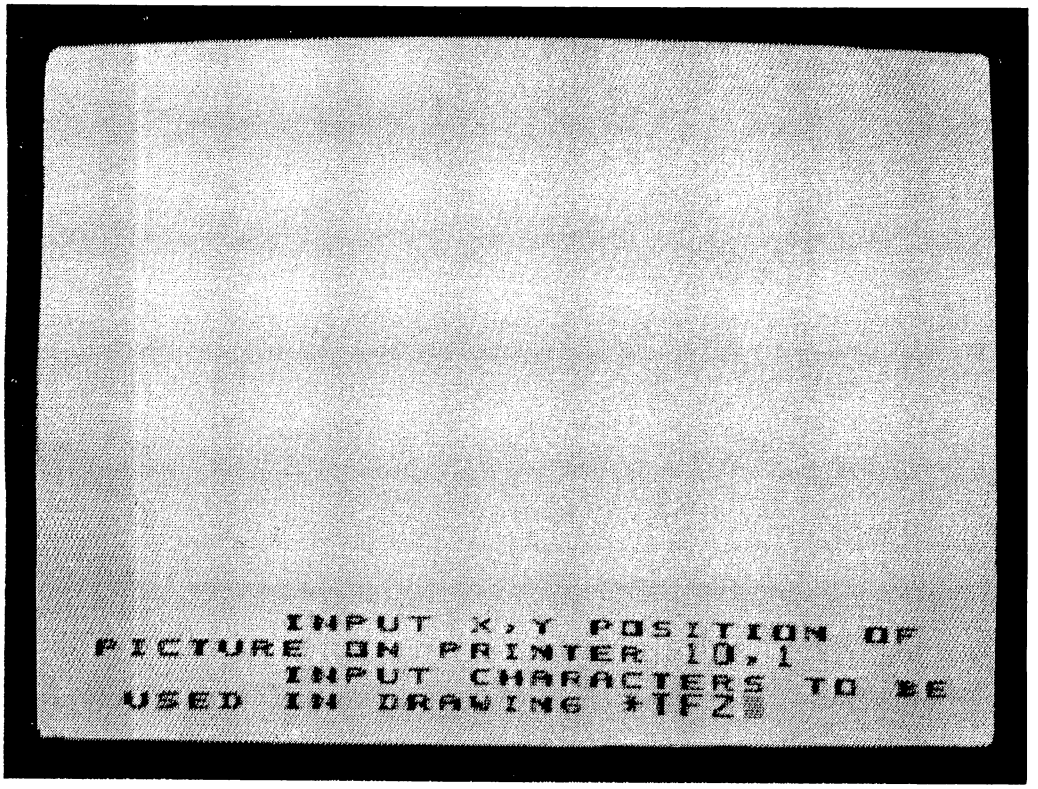
Problem 5.6

Draw the Bentley College logo on the line printer.

Solution

```
10 ! filename:"gr5p6"
20 !  purpose: draw Bentley College logo on line printer
30 !   author: jpg, jdr & tfz 10/83
40 !
50 DIM P$(64):: CALL CLEAR
60 FOR I=0 TO 63
70 P$(I)=RPT$(" ",64):: !blanks
80 NEXT I
90 INPUT "      input x,y position of picture on printer ":XINC,YINC
100 INPUT "      input characters to be used in drawing ":CHAR$
110 CHAR$=SEG$(CHAR$,1,10)
120 GOSUB 1000 :: !  Draw the picture
130 CALL KEY(0,N,S):: IF S=0 THEN 130 ELSE 9999
1000 !  Subroutine to draw picture on screen
1010 CALL CLEAR
1020 READ N :: !  Number of straight lines to draw in picture
1025 OPEN #1:"RS232"
1030 FOR L=1 TO N
1040 READ E,C,D
1050 X1=INT(C/100):: Y1=C-100*X1
1060 X2=INT(D/100):: Y2=D-100*X2
1070 X1=X1+XINC :: Y1=Y1+YINC :: !  Relocate coordinates
1080 X2=X2+XINC :: Y2=Y2+YINC
1090 IF X1>63 OR X2>63 OR Y1>63 OR Y2>63 THEN PRINT "picture will not fit, canno
t continue" :: RETURN
```

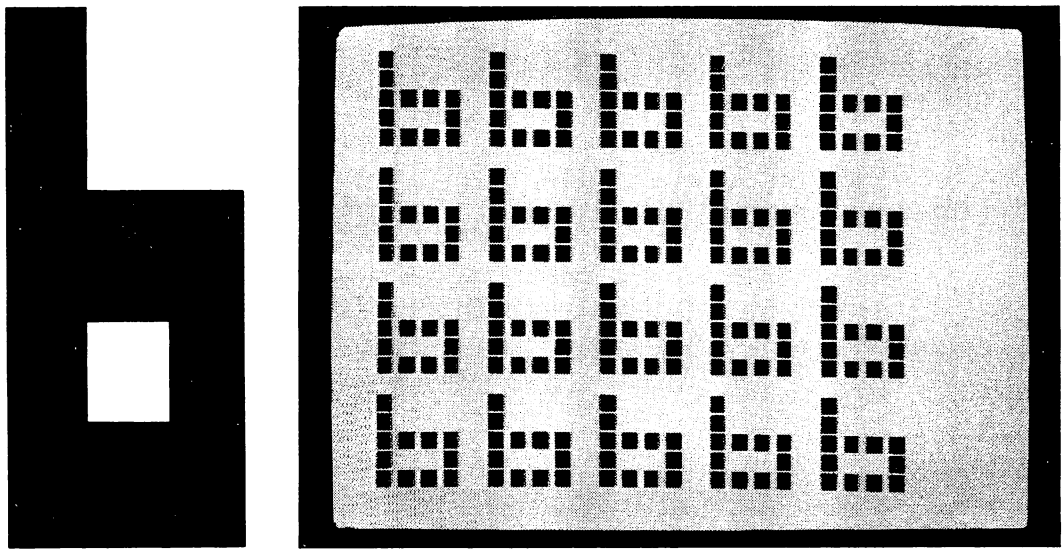
```
1100 IF X1=X2 THEN 1260
1110 IF Y1=Y2 THEN 1300
1120 M=(Y2-Y1)/X2-X1):: !  Calculate slope of line
1130 B=Y1-M*X1 :: !  Calculate y-intercept
1140 IF M>0 THEN IF M<=1 THEN 1210 ELSE 1160 ELSE IF M>=-1 THEN 1210 ELSE 1150
1150 T=Y1 :: Y1=Y2 :: Y2=T :: !  Slope between -1 and -infinity
1160 FOR J=Y1 TO Y2 :: !  Slope between 1 and infinity
1170 I=INT((J-B)/M+.5)
1180 P$(J)=SEG$(P$(J),1,I)&SEG$(CHAR$,E+1,1)&SEG$(P$(J),I+2,LEN(P$(J)))
1190 NEXT J
1200 GOTO 1330
1210 FOR I=X1 TO X2 :: !  Slope between -1 and 1
1220 J=INT(M*I+B+.5)
1230 P$(J)=SEG$(P$(J),1,I)&SEG$(CHAR$,E+1,1)&SEG$(P$(J),I+2,LEN(P$(J)))
1240 NEXT I
1250 GOTO 1330
1260 FOR J=Y1 TO Y2 :: !  No slope, vertical line
1270 P$(J)=SEG$(P$(J),1,X1)&SEG$(CHAR$,E+1,1)&SEG$(P$(J),X1+2,LEN(P$(J)))
1280 NEXT J
1290 GOTO 1330
1300 FOR I=X1 TO X2 :: !  Zero slope, horizontal line
1310 P$(Y1)=SEG$(P$(Y1),1,I)&SEG$(CHAR$,E+1,1)&SEG$(P$(Y1),I+2,LEN(P$(Y1)))
1320 NEXT I
1330 NEXT L
1340 FOR J=0 TO 63 :: PRINT #1:P$(J):: NEXT J
1350 RETURN
2000 DATA 31,0,0,4700,0,1,14,0,101,114,0,4601,4614,0,4701,4714
2010 DATA 0,15,4715,0,1005,1705,0,806,1906,0,607,612,0,707,712
2020 DATA 0,807,812,0,907,912,0,1807,1812,0,1907,1912
2030 DATA 0,2007,2012,0,2107,2112,0,613,2113,0,3002,3702
2040 DATA 0,2803,3903,0,2604,2612,0,2704,2712,0,2804,2812
2050 DATA 0,2904,2912,0,3804,3812,0,3904,3912,0,4004,4012
2060 DATA 0,4104,4112,0,2613,4113,1,302,302,2,402,402,3,502,502
9000 CLOSE #1
9999 END
```




```
**********************************************
**                                          **
** TFZ                        ********       **
**                          ************    **
**                       ****        ****   **
**        ********       ****        ****   **
**      ************     ****        ****   **
**    ****        ****   ****        ****   **
**    ****        ****   ****        ****   **
**    ****        ****   ****        ****   **
**    ****        ****   ****        ****   **
**    ****        ****   ****        ****   **
**    ****        ****   ****        ****   **
**    ****************   ****************   **
**                                          **
**********************************************
```

Discussion

- Notice that this is program GR5P5 with a few minor changes. The DISPLAY AT's were changed to LET statements using the SEG$ function on the *left* of the equal sign. A simple routine to PRINT the strings at the subroutine's end furnishes the output to the printer.



Character Graphics

Character Graphics techniques rely on the use of a few advanced BASIC commands and the graphic character set. This chapter will explore the full range of instructions in the TI–99/4A's BASIC that is needed for this type of graphics.

DISPLAY AT

The TI–99/4A video screen is 27 columns wide and 24 rows deep. Each of the 648 screen positions is addressable by row and column with the DISPLAY AT instruction. The top left position is in row one and column one. The cursor is positioned to this location with the instruction DISPLAY AT(1,1). The bottom right of the screen can be reached with a DISPLAY AT(24,27) instruction.

The DISPLAY AT instruction positions the cursor at the screen position (row 1, column 1 to row 24, column 27). Then the next PRINT instruction that is executed in the program will begin printing at that location.

Instruction	Output
`10 DISPLAY AT(1,27):"X"`	X at top of screen
`20 DISPLAY AT(2,13):"ZOT"`	ZOT centered on second line
`30 DISPLAY AT(13,13):"ZOTZOT"`	ZOTZOT centered on screen
`40 DISPLAY AT(24,1):"Z";`	Z at bottom left of screen

If the DATA that follows a DISPLAY AT instruction ends with no punctuation, the cursor returns to the beginning of the next line, and this may produce undesirable results. If the address is in row 24, the string points on the last line of the screen, then the screen scrolls one line. The effect of a DISPLAY AT(24,1):X becomes the same as a DISPLAY AT(23,1):Y, which is not what was intended.

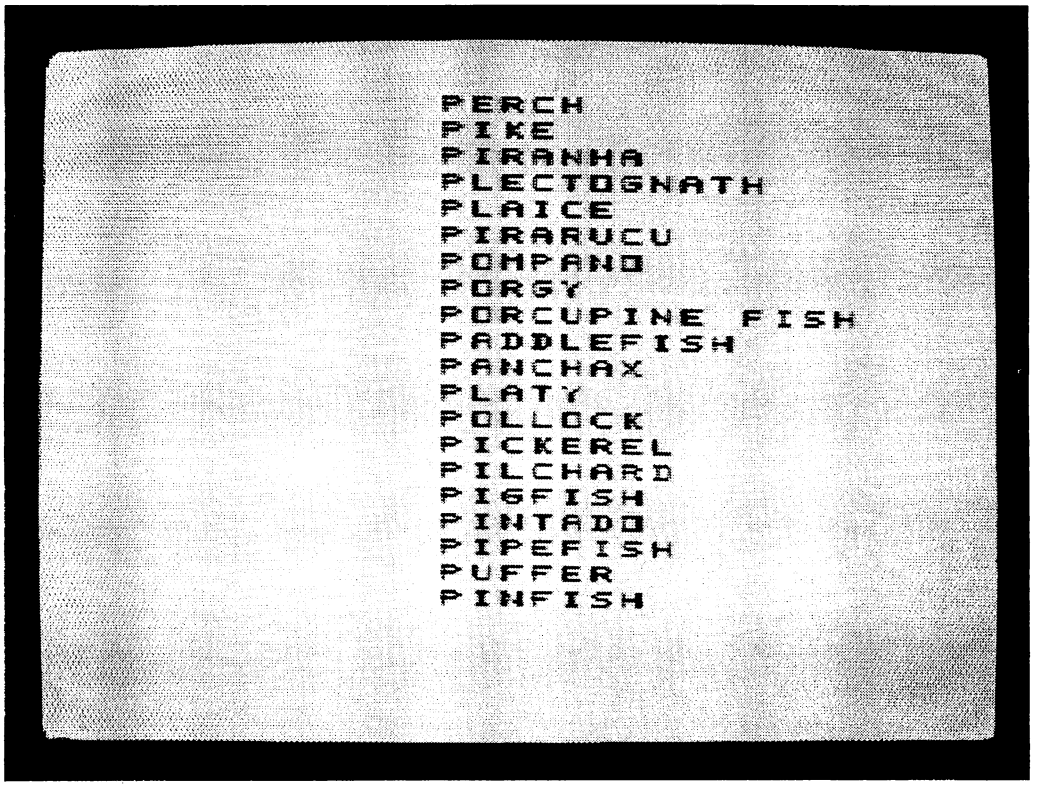
If the DATA ends with a semicolon (;), the result is predictable. We recommend that you always end all DATA with a semicolon when they follow a DISPLAY AT.

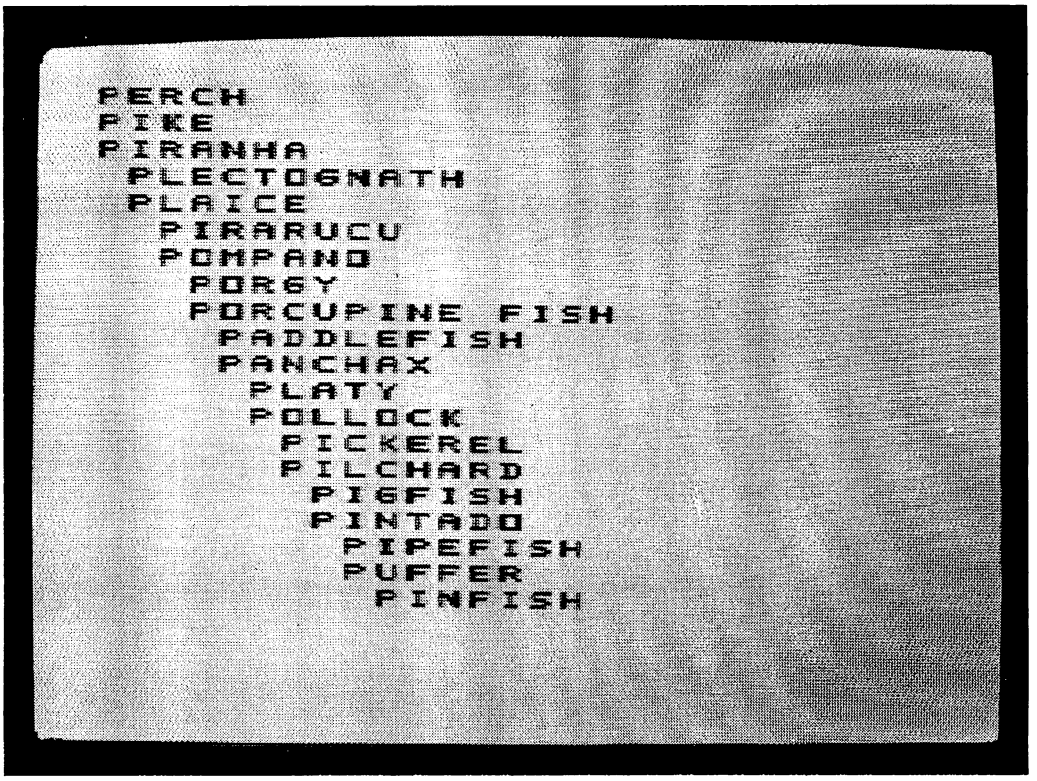
Problem 6.1

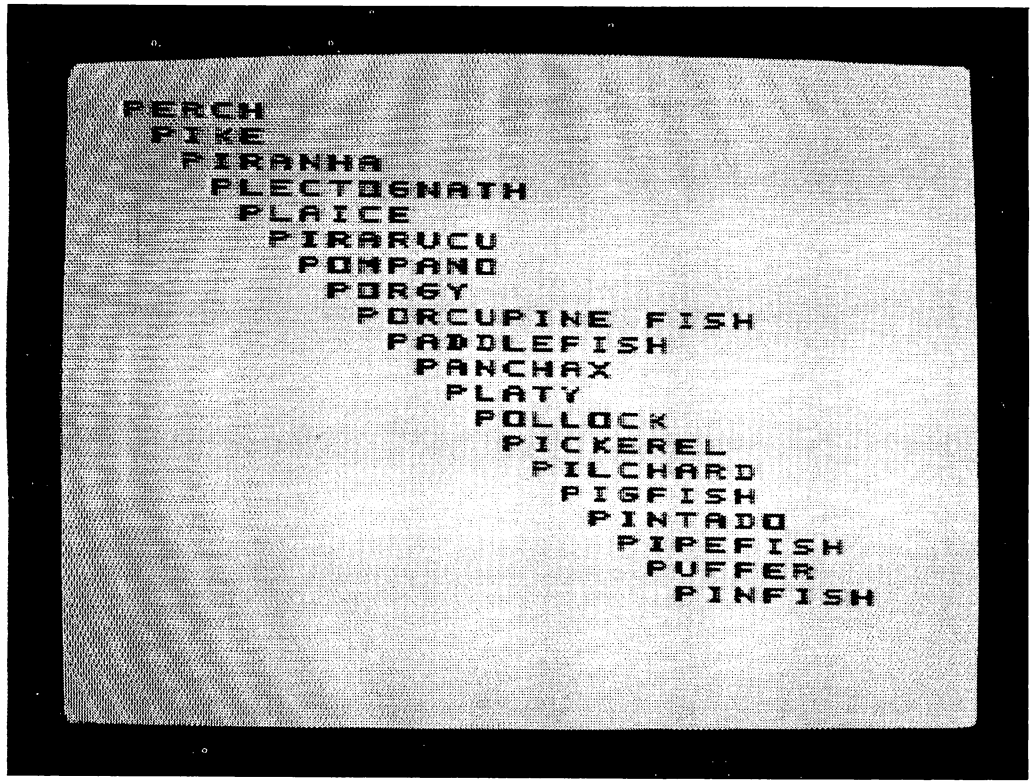
List a table in a variety of ways using the DISPLAY AT statement.

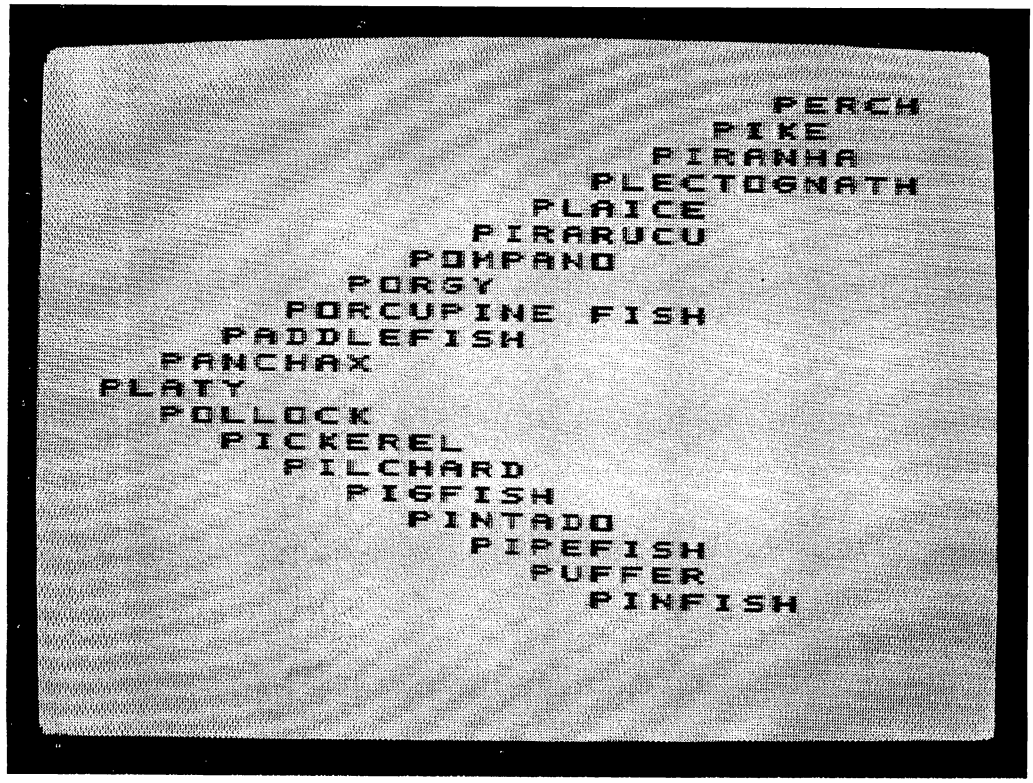
Solution

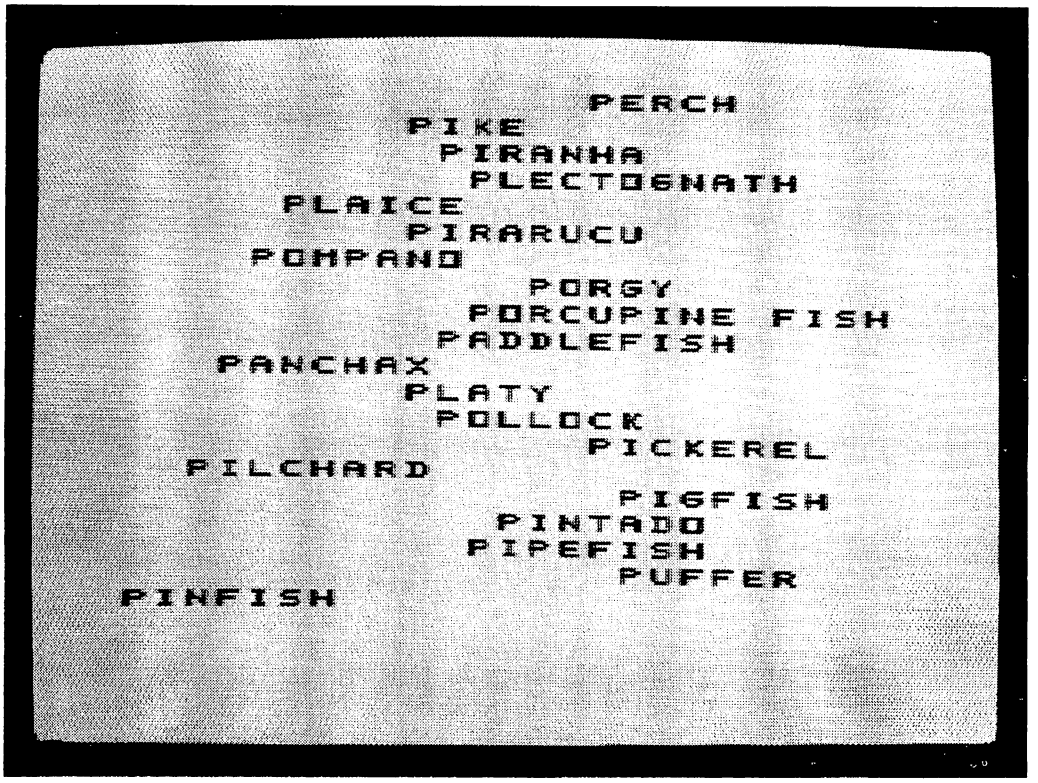
```
10 ! filename:"gr6p1"
20 !  purpose: output table in sundry ways using DISPLAY AT statement
30 !   author: jpg, jdr & tfz 10/83
40 !
50 DIM P$(50)
60 READ N :: FOR I=1 TO N :: READ P$(I):: NEXT I
70 RANDOMIZE :: CALL CLEAR
80 FOR I=1 TO N
90 DISPLAY AT(I,12):P$(I)
100 NEXT I :: GOSUB 1000 :: !pause and clear screen
110 FOR I=1 TO 20
120 DISPLAY AT(I,INT(.5*I)):P$(I)
130 NEXT I :: GOSUB 1000 :: !pause & clear
140 FOR I=1 TO 20
150 DISPLAY AT(I,I):P$(I)
160 NEXT I :: GOSUB 1000 :: !pause & clear
170 FOR I=1 TO 20
180 DISPLAY AT(I,2*ABS(12-I)+1):P$(I)
190 NEXT I :: GOSUB 1000 :: !pause & clear
200 FOR I=1 TO 20
210 DISPLAY AT(I,1+INT(18*RND)):P$(I)
220 NEXT I :: GOSUB 1000 :: !pause & clear
230 FOR I=1 TO 20
240 DISPLAY AT(I,20-LEN(P$(I))):P$(I)
250 NEXT I :: GOSUB 1000 :: !pause & clear
260 STOP
270 DATA 20,perch,pike,piranha,plectognath,plaice
280 DATA pirarucu,pompano,porgy,porcupine fish
290 DATA paddlefish,panchax,platy,pollock,pickerel
300 DATA pilchard,pigfish,pintado,pipefish,puffer
310 DATA pinfish,pumpkinseed,parrotfish,phlounder
1000 !pause and clear screen subroutine
1010 CALL KEY(0,N,S):: IF S=0 THEN 1010
1020 CALL CLEAR
1030 RETURN
9999 END
```

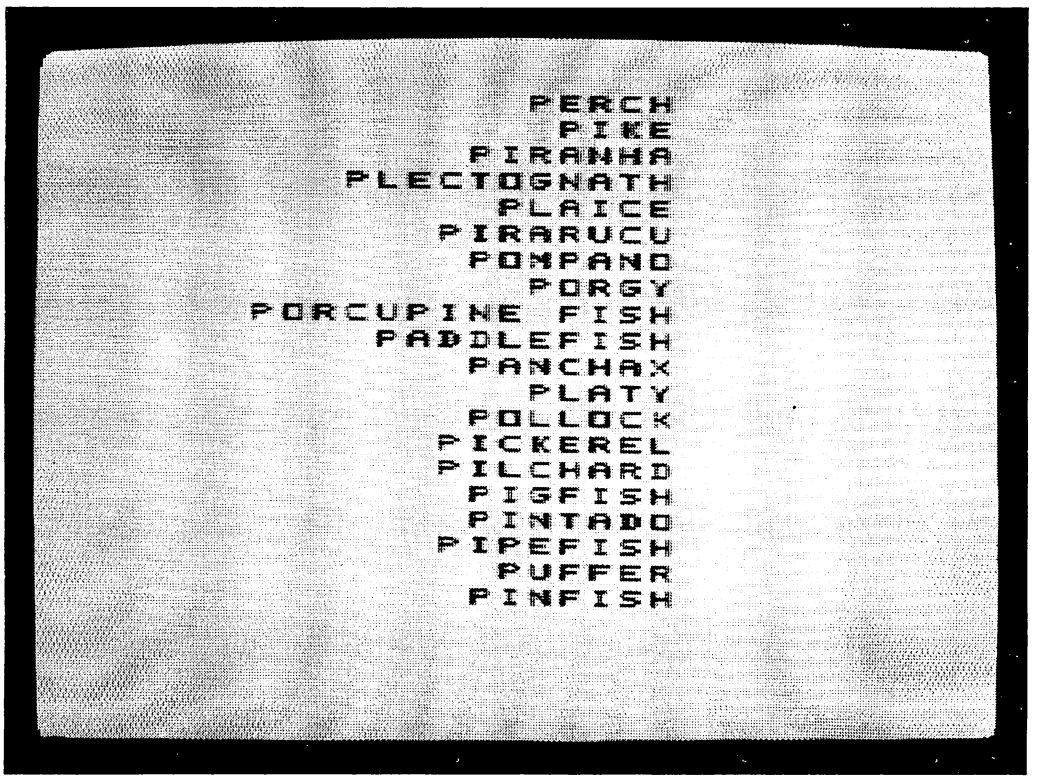

PERCH
PIKE
PIRANHA
PLECTOGNATH
PLAICE
PIRARUCU
POMPANO
PORGY
PORCUPINE FISH
PADDLEFISH
PANCHAX
PLATY
POLLOCK
PICKEREL
PILCHARD
PIGFISH
PINTADO
PIPEFISH
PUFFER
PINFISH


PERCH
PIKE
PIRANHA
PLECTOGNATH
PLAICE
PIRARUCU
POMPANO
PORGY
PORCUPINE FISH
PADDLEFISH
PANCHAX
PLATY
POLLOCK
PICKEREL
PILCHARD
PIGFISH
PINTADO
PIPEFISH
PUFFER
PINFISH


PERCH
PIKE
PIRANHA
PLECTOGNATH
PLAICE
PIRARUCU
POMPANO
PORGY
PORCUPINE FISH
PADDLEFISH
PANCHAX
PLATY
POLLOCK
PICKEREL
PILCHARD
PIGFISH
PINTADO
PIPEFISH
PUFFER
PINFISH


PERCH
PIKE
PIRANHA
PLECTOGNATH
PLAICE
PIRARUCU
POMPANO
PORGY
PORCUPINE FISH
PADDLEFISH
PANCHAX
PLATY
POLLOCK
PICKEREL
PILCHARD
PIGFISH
PINTADO
PIPEFISH
PUFFER
PINFISH


PERCH
PIKE
PIRANHA
PLECTOGNATH
PLAICE
PIRARUCU
POMPANO
PORGY
PORCUPINE FISH
PADDLEFISH
PANCHAX
PLATY
POLLOCK
PICKEREL
PILCHARD
PIGFISH
PINTADO
PIPEFISH
PUFFER
PINFISH


PERCH
PIKE
PIRANHA
PLECTOGNATH
PLAICE
PIRARUCU
POMPANO
PORGY
PORCUPINE FISH
PADDLEFISH
PANCHAX
PLATY
POLLOCK
PICKEREL
PILCHARD
PIGFISH
PINTADO
PIPEFISH
PUFFER
PINFISH

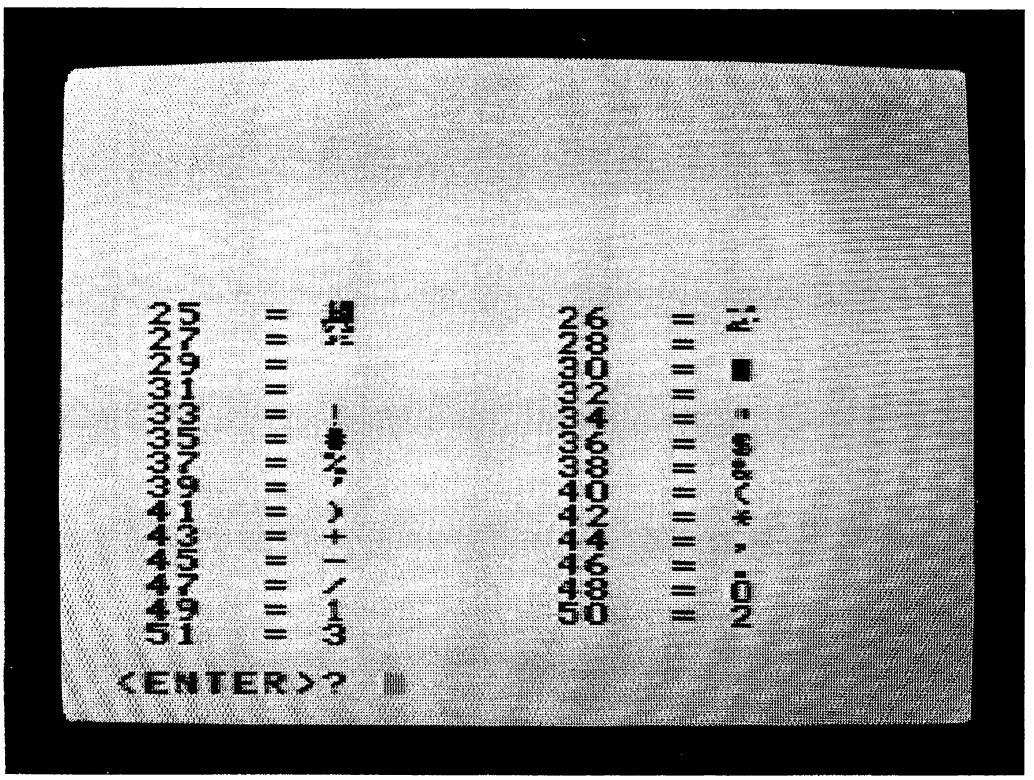
RPT$

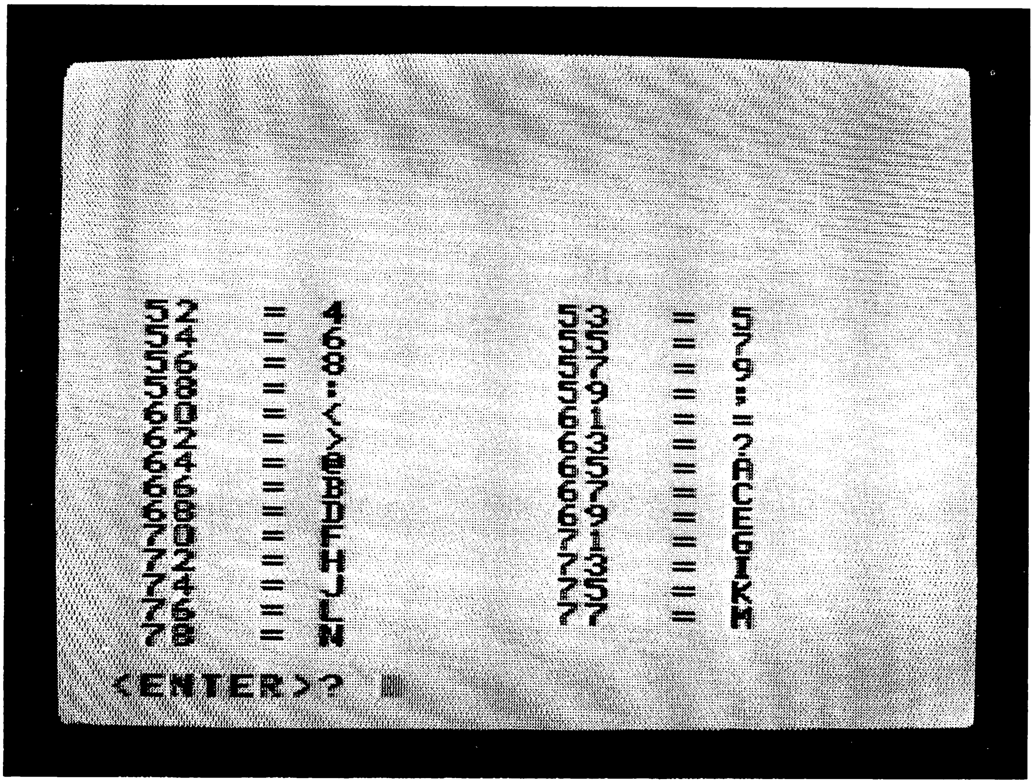
A BASIC function which is useful in graphing is the RPT$ function. This function has two arguments: The first is the number of characters desired, up to 255, and the second is the character itself.

Instruction	Output
`10 RPT$ ("*",10)`	**********
`20 RPT$ (CHR$(65),4)`	AAAAA
`30 RPT$ (CHR$(13),8)`	Cursor moves down 8 lines, positions itself at left of line. CHR$(13) is a carriage/ cursor return which is the character produced by the ENTER key.
`40 DISPLAY AT(24,1): RPT$("Z",28);`	ZZZ . . . ZZZ (80 of them) on the last line of the screen
`50 A=128::B$="9"::` `RPT$(B$,A)`	Two rows of 9s

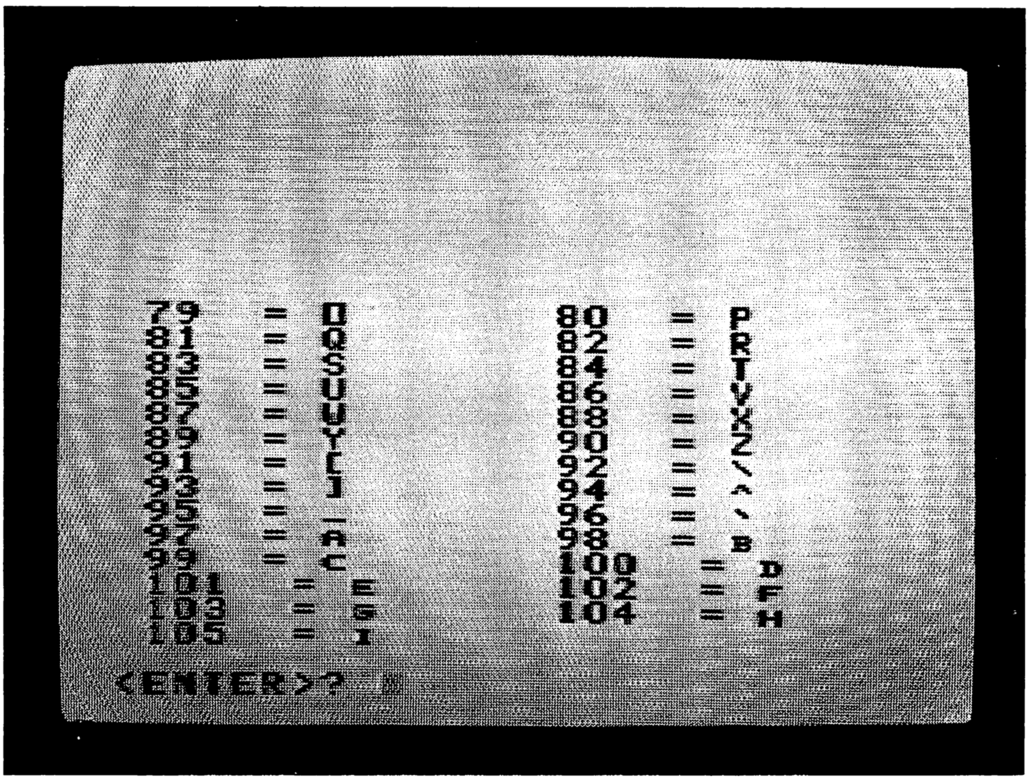
The Character Set

A look at the following pictures reveals that the TI–99/4A has a total of 96 possible characters, and the effect of each can be displayed by using the instruction PRINT CHR$(N) where N is a number from 30 to 126. A few of these values have no effect on the screen but most do.





52 = 4
53 = 5
54 = 6
55 = 7
56 = 8
57 = 9
58 = :
59 = ;
60 = <
61 = =
62 = >
63 = ?
64 = @
65 = A
66 = B
67 = C
68 = D
69 = E
70 = F
71 = G
72 = H
73 = I
74 = J
75 = K
76 = L
77 = M
78 = N
<ENTER>?



79 = O
80 = P
81 = Q
82 = R
83 = S
84 = T
85 = U
86 = V
87 = W
88 = X
89 = Y
90 = Z
91 = [
92 = \
93 =]
94 = ^
95 = _
96 = `
97 = A
98 = B
99 = C
100 = D
101 = E
102 = F
103 = G
104 = H
105 = I
<ENTER>?

Problem 6.2 Display the graphics characters on the video screen.

Solution

```
10 ! filename:"gr6p2"
20 !  purpose: display graphic characters
30 !   author: jpg, jdr & tfz/83
40 !
50 CALL CLEAR
60 READ B,E
70 IF B<0 THEN 130
80 FOR I=B TO E
90 PRINT I;" = ";CHR$(I),
100 NEXT I
110 PRINT :: PRINT :: PRINT "<ENTER>";:: INPUT R$ :: CALL CLEAR
120 GOTO 60
130 STOP
140 DATA 25,51,52,78,79,105,106,126,-1,-1
150 END
```

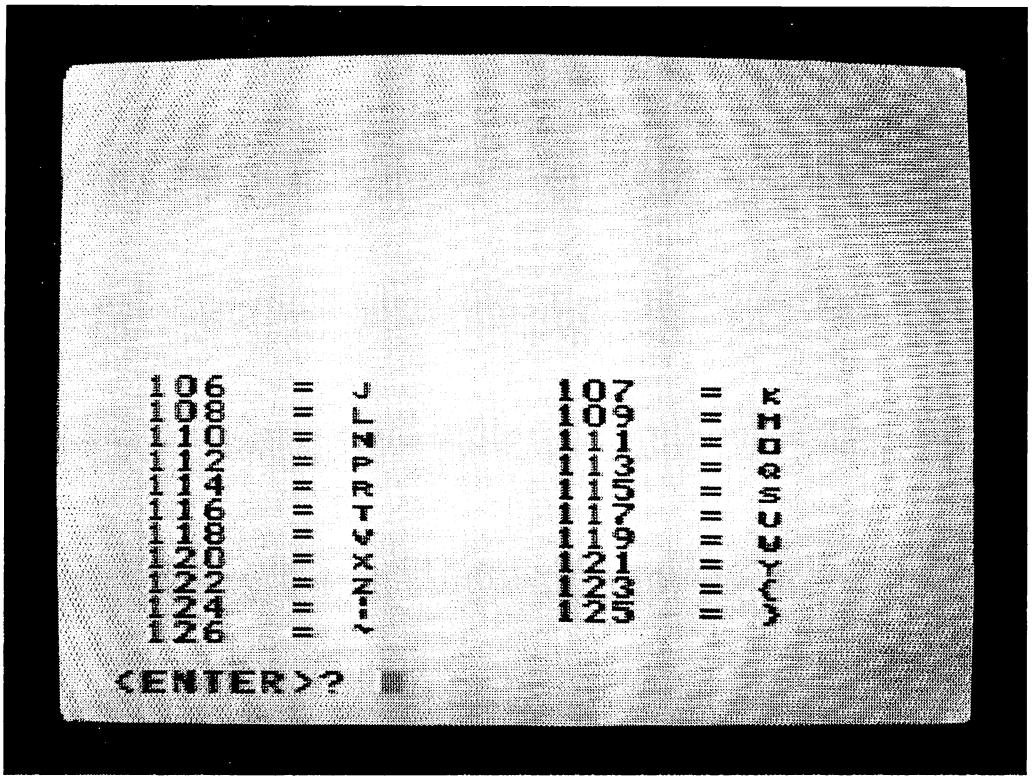
Suggestions

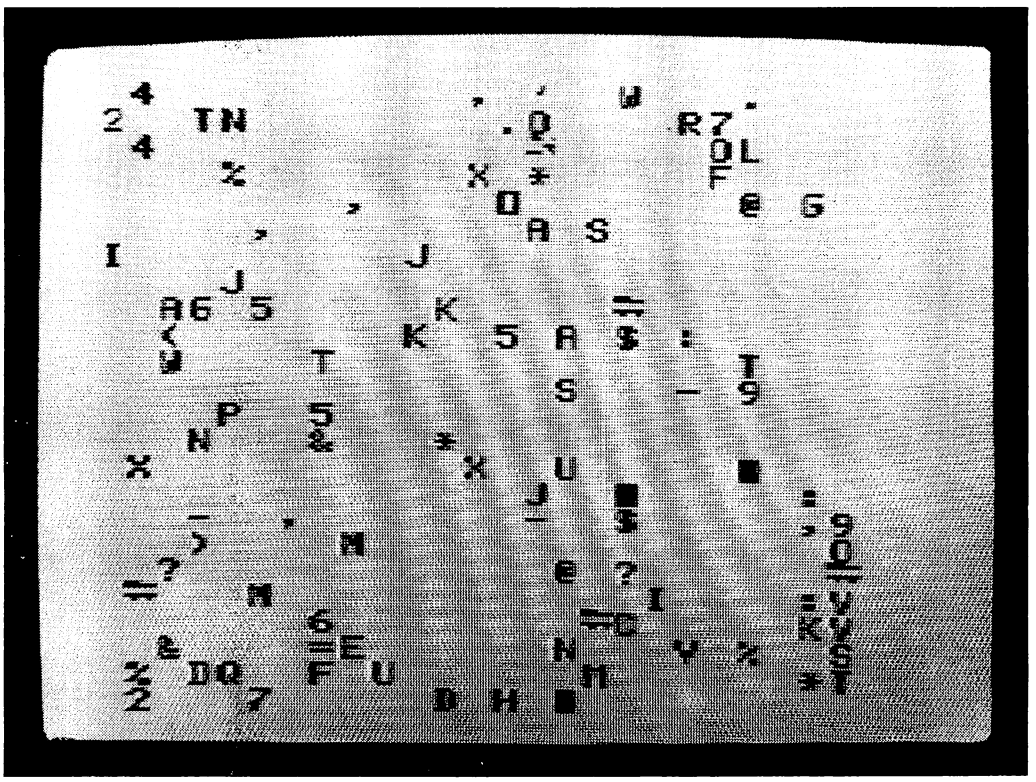
- Modify the DATA statement to produce different sequences of characters.
- Modify the program to arrange the character codes vertically on the screen.

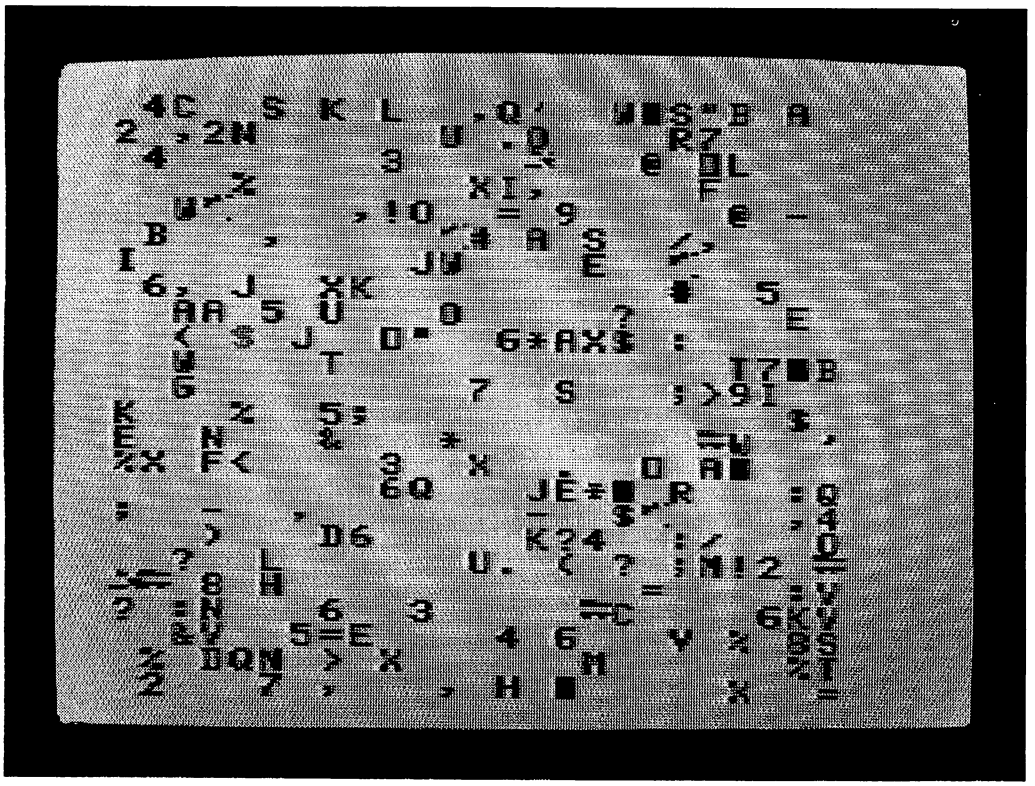
Problem 6.3 Create a dynamic visual display using the TI–99/4A graphics characters.

Solution

```
10 ! filename:"gr6p3"
20 !  purpose: random pattern of graphic characters
30 !   author: jpg, jdr & tfz 10/83
40 !
50 DIM G$(126):: CALL CLEAR :: K=0
60 FOR I=1 TO 4
70 READ B,E
80 FOR J=B TO E :: K=K+1 :: G$(K)=CHR$(J):: NEXT J
90 NEXT I
100 FOR I=1 TO 1024
110 J=1+INT(25*RND):: K=1+INT(24*RND)
120 L=1+INT(64*RND):: DISPLAY AT(K,J):G$(L);
130 NEXT I
140 DATA 25,51,52,78,79,105,106,126
150 END
```


<ENTER>?





Discussion

- A very simple program can create an attractive visual display.

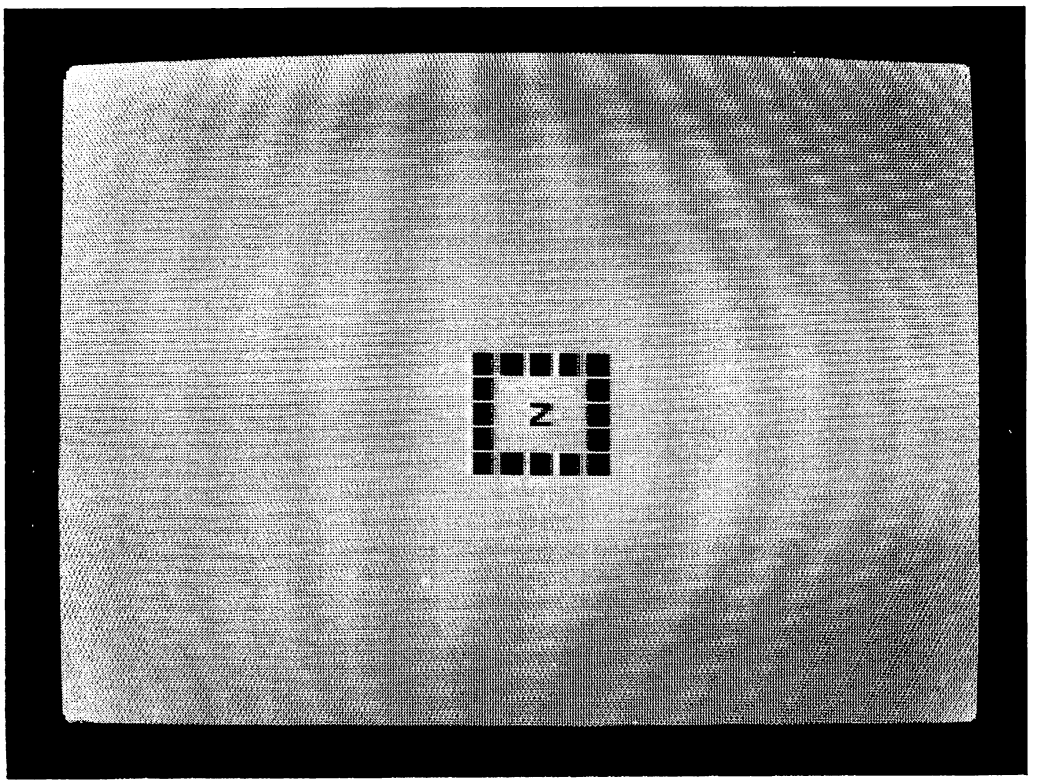
Suggestions

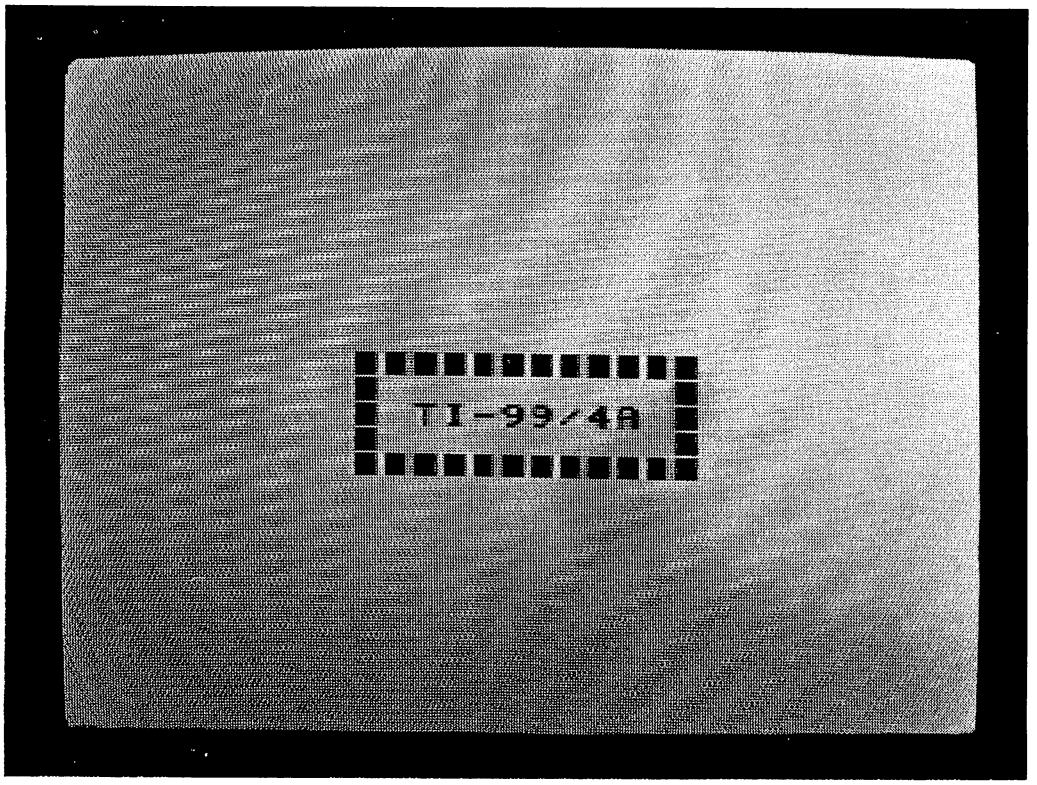
- Modify the program so that the display uses double-wide graphics characters.
- Modify the program so that it outputs all TI–99/4A characters in the display.

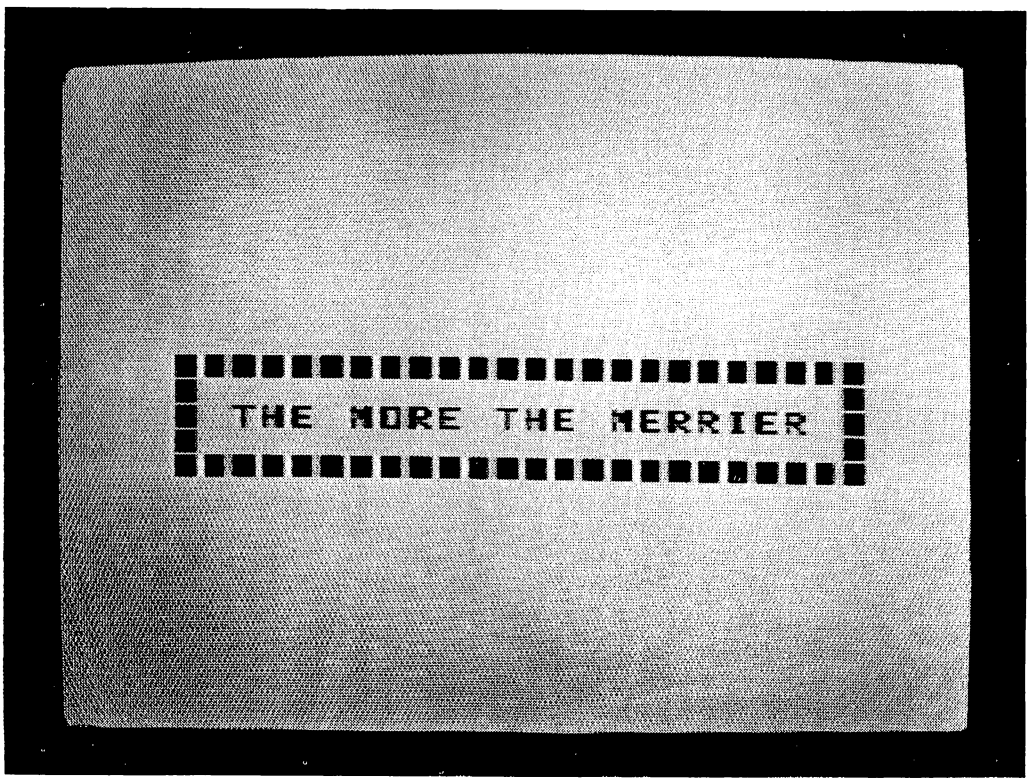
Problem 6.4 Write a program that will put a box around a message near the center of the screen.

Solution

```
10 ! filename:"gr6p4"
20 !  purpose: place a message in a box
30 !   author: jpg, jdr & tfz/83
40 !
50 CALL CLEAR
60 PRINT "input message with less than 20 characters"
70 INPUT T$ :: IF LEN(T$)>20 THEN PRINT "Message too long" :: PRINT :: GOTO 60
80 T$=" "&T$&" "
90 CALL CLEAR
100 L1=LEN(T$):: L2=INT(32-L1)/2-1
110 CALL HCHAR(11,L2+2,30,L1)
120 CALL HCHAR(15,L2+2,30,L1)
130 FOR I=1 TO 3
140 CALL VCHAR(11,L2+1,30,5)
150 CALL VCHAR(11,L1+L2+2,30,5)
160 NEXT I
170 DISPLAY AT(13,L2):T$;
180 END
```




TI-99/4A


THE MORE THE MERRIER

Discussion

- The variable L points to the position of the screen where the message will be printed.
- HCHAR and VCHAR are used to draw the horizontal and vertical bars, respectively. The first two numbers indicate placement, the next number is the character number, and the last number indicates how many times the character is to be displayed.
- Notice how the values of the variables L1 and L2 are used to position the box and message.

Suggestions

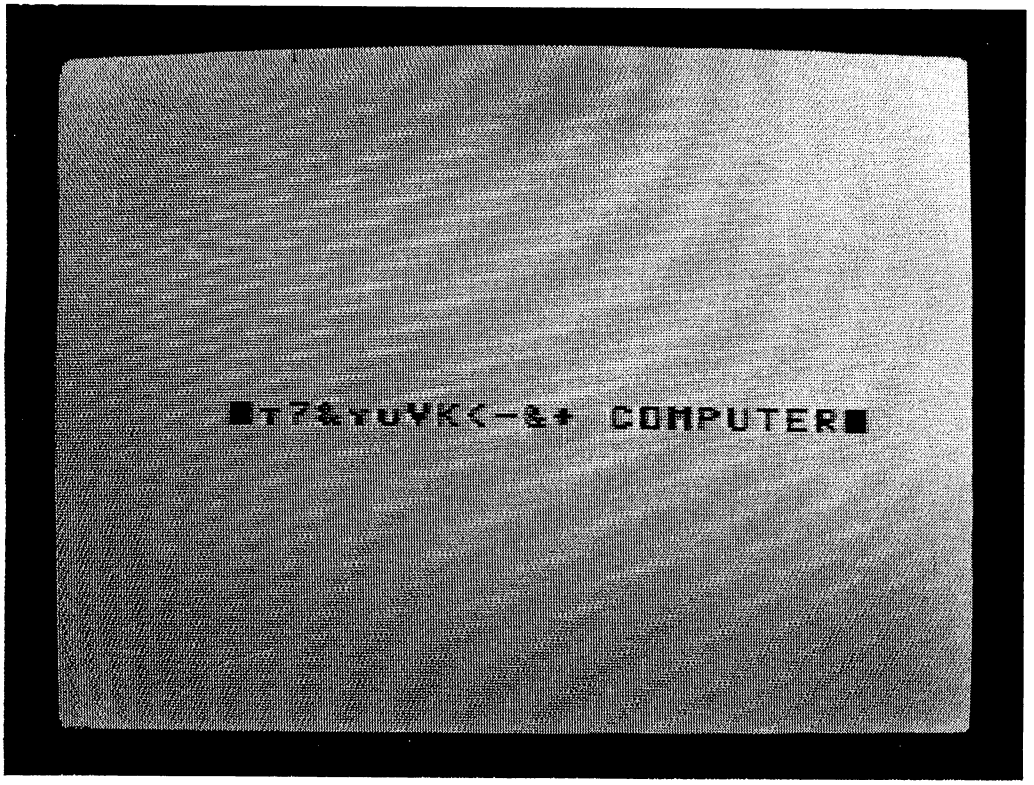
- Write the boxing portion of the program as a subroutine that can be called to box and center a message.
- Generalize the program so that the user can specify the position of the boxed message.

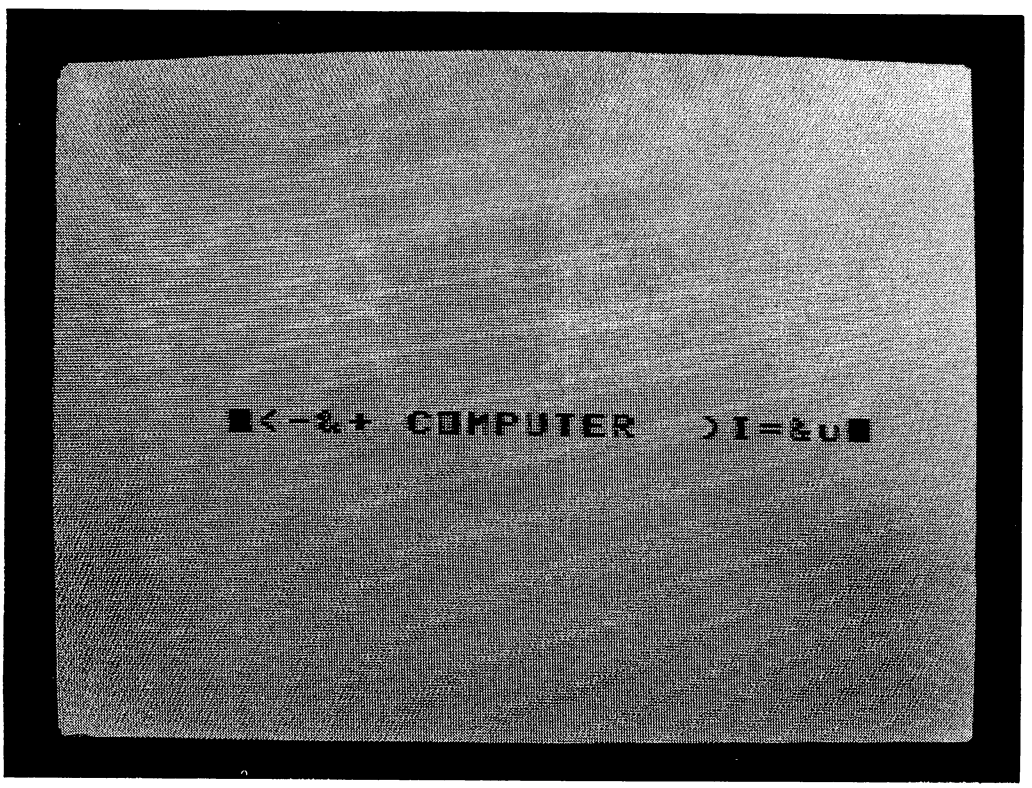
Problem 6.5

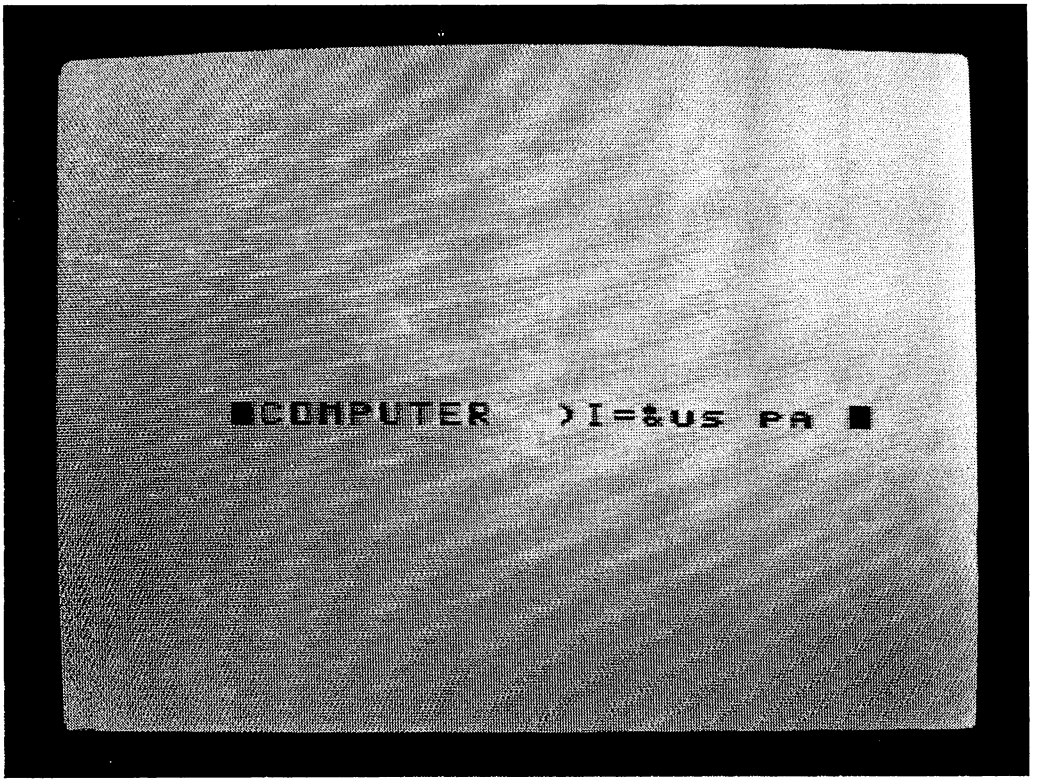
Write a program that will display a message as a moving banner similar to the flashing light messages appearing at night on the side of a blimp.

Solution

```
10 ! filename:"gr6p5"
20 !  purpose: moving banner of characters
30 !   author: jpg, jdr & tfz/83
40 !
50 CALL CLEAR
60 INPUT "enter message to display":B$
70 C$="" :: D$="" :: Z=150 :: !Z is speed factor
80 FOR I=1 TO 15
90 C$=C$&CHR$(RND*96+30):: D$=D$&CHR$(RND*96+30)
100 NEXT I
110 CALL CLEAR
120 B$=RPT$(" ",6)&C$&" "&B$&" "&D$&RPT$(" ",6)
130 DISPLAY AT(13,5):CHR$(30);
140 DISPLAY AT(13,26):CHR$(30);
150 I=1
160 FOR J=1 TO 1000
170 DISPLAY AT(13,6):SEG$(B$,I,20);
180 I=I+1 :: IF I>LEN(B$)-20 THEN I=1
190 FOR K=1 TO Z :: NEXT K :: !pause
200 NEXT J
210 END
```


COMPUTER


COMPUTER



Discussion

- 15 randomly selected graphics characters are affixed to the beginning and the end of the message to be displayed.
- 19 blanks are affixed to the beginning and end of the string to be displayed.
- A 20 character wide window is provided for the message to move through and be displayed.
- The motion is obtained by using the SEG$ function to select the 20 characters to be printed after the DISPLAY AT statement.
- A delay loop is needed to slow down the TI–99/4A so that the message can be read.

Suggestions

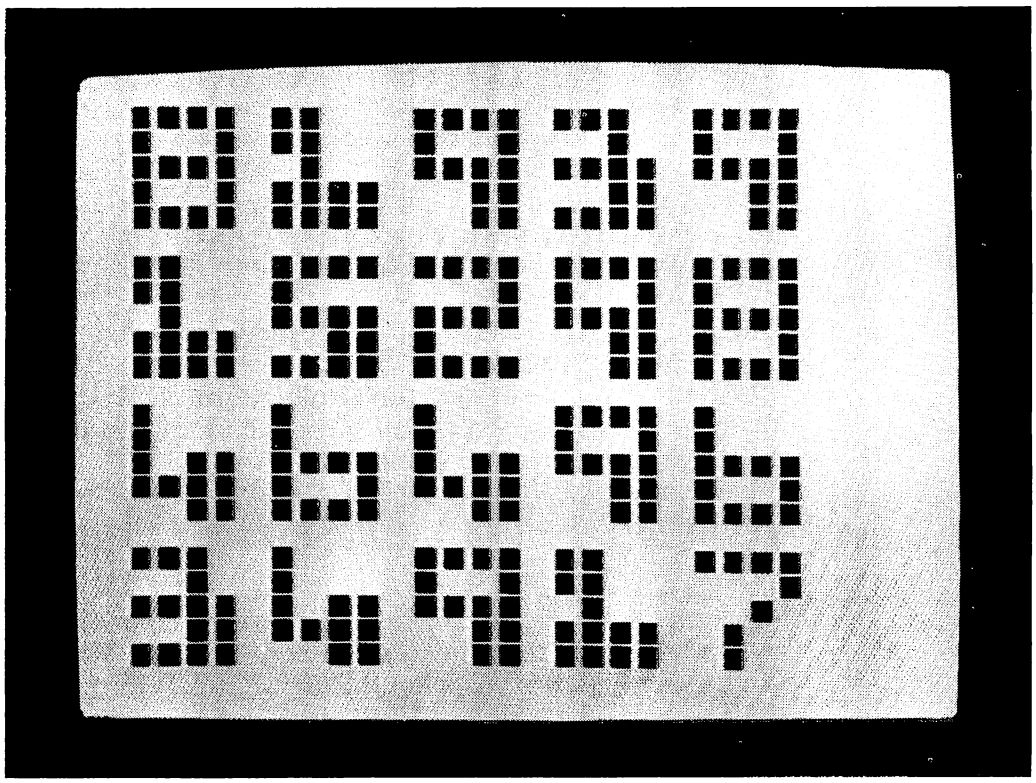
- Convert the program to a subroutine that can be called to create a moving banner out of a message.
- Modify the program so that the width of the banner is specified by the user of the program.
- Modify the program so that the position of the banner can be specified by the user of the program.
- Convert the program to produce a double-wide banner.

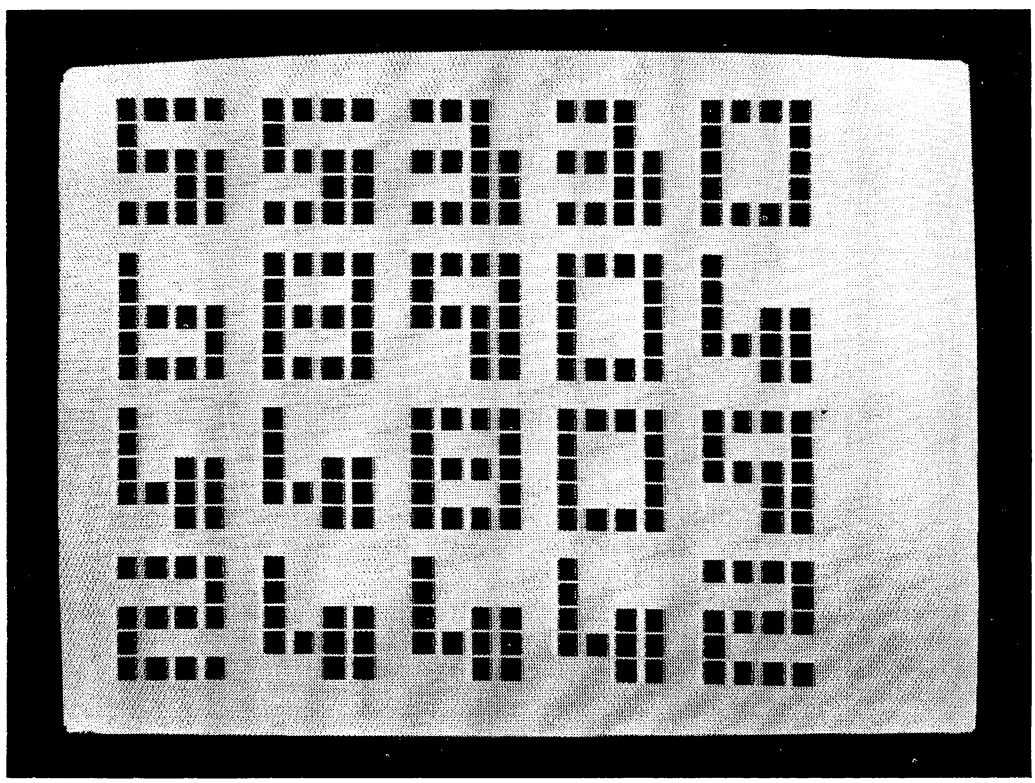
Problem 6.6

Produce a screen full of oversize random digits.

Solution

```
10 ! filename:"gr6p6"
20 !  purpose: produce screen full of large digits
30 !   author: jpg, jdr & tfz/83
40 !
50 CALL CLEAR :: RANDOMIZE
60 DIM G$(10,5)
70 FOR I=1 TO 10
80 FOR J=1 TO 5
90 A$=""
100 READ X
110 FOR K=1 TO 4
120 IF X/10<>INT(X/10)THEN A$=CHR$(30)&A$ ELSE A$=" "&A$
130 X=INT(X/10)
140 NEXT K
150 G$(I,J)=A$
160 NEXT J
170 NEXT I
180 DATA 1111,1001,1001,1001,1111
190 DATA 1100,1100,0100,1111,1111
200 DATA 1111,0001,1111,1000,1111
210 DATA 1110,0010,1111,0011,1111
220 DATA 1000,1000,1011,1111,0011
230 DATA 1111,1000,1111,0011,1111
240 DATA 1000,1000,1111,1001,1111
250 DATA 1111,0001,0010,0100,0100
260 DATA 1111,1001,1111,1001,1111
270 DATA 1111,1001,1111,0011,0011
280 FOR X=1 TO 25 STEP 5
290 FOR Y=1 TO 20 STEP 6
300 D=INT(10*RND)
310 GOSUB 1000 :: !display large digit
320 NEXT Y
330 NEXT X
340 CALL KEY(0,N,S):: IF S=0 THEN 340 ELSE 9999
1000 !  Display large digit D at X,Y
1010 FOR I8=0 TO 4
1020 DISPLAY AT(Y+I8,X):G$(D+1,I8+1);
1040 NEXT I8
1050 RETURN
9999 END
```





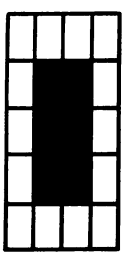
Discussion

- This program couples the table-driven technique with character graphics.

- The enlarged digit is constructed using one graphics character, the large blip produced when CHR$(30) is printed. For example, the first five numbers in the DATA statements represent the digit 0. The 1111, 1001, 1001, 1001, and 1111 indicate the presence or absence of the graphics character, the blip, that will form the digit.

  The first one, 1111, is the top of the zero. Each digit 1 says in effect, "Print a blip" and each digit 0 says, "Leave a blank." The next three, 1001, form the body of the rectangular digit 0. Each of these will be transformed by the program into the commands, "Print a blip," "Leave a blank," "Leave a blank," and "Print a blip." The last DATA item for the digit 0 is 1111, which closes the loop.

  When the program prints the digit, it does so by printing the five strings in the array G$ that correspond to the five 4-character lines that make up the digit. G$(1,1) through G$(1,5) holds the digit 0. G$(2,1) through G$(2,5) holds the digit 1, and so on until G$(10,1) through G$(10,5), which is used for the digit 9. Study the table below:

DATA G$ characters I8	What is printed
1111 219,219,219,219 0	
1001 219,“ ”,“ ”,219 1	
1001 219,“ ”,“ ”,219 2	
1001 219,“ ”,“ ”,219 3	
1111 219,219,219,219 4	

Suggestions

- Modify the DATA statements so that strange symbols instead of digits are displayed.

- Write a program that uses the ideas of the oversize digits to output the value of a numeric variable in oversize form at the center of the screen.

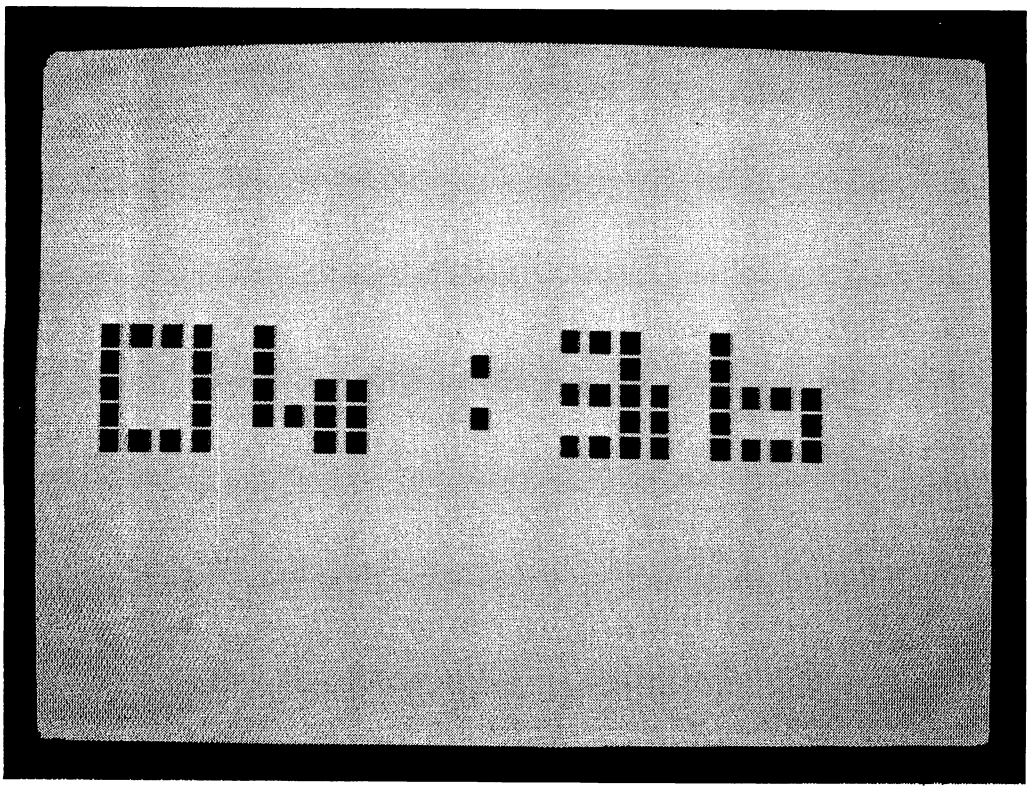
- Modify the program so that the oversize digits disappear one at a time from left to right, top to bottom.

Problem 6.7

Write a program that will display a large-digit digital clock showing hour and minutes in the lower left-hand corner of the screen.

Solution

```
10 ! filename:"gr6p7"
20 !  purpose: digital clock in lower left of screen
30 !   author: jpg, jdr, tfz & mfz 10/83
40 !
50 CALL CLEAR
60 DIM G$(10,5),T(5)
70 FOR I=1 TO 10
80 FOR J=1 TO 5
90 A$=""
100 READ X
110 FOR K=1 TO 4
120 IF X/10<>INT(X/10)THEN A$=CHR$(30)&A$ ELSE A$=" "&A$
130 X=INT(X/10)
140 NEXT K
150 G$(I,J)=A$
160 NEXT J :: NEXT I
180 DATA 1111,1001,1001,1001,1111
190 DATA 1100,1100,0100,1111,1111
200 DATA 1111,0001,1111,1000,1111
210 DATA 1110,0010,1111,0011,1111
220 DATA 1000,1000,1011,1111,0011
230 DATA 1111,1000,1111,0011,1111
240 DATA 1000,1000,1111,1001,1111
250 DATA 1111,0001,0010,0100,0100
260 DATA 1111,1001,1111,1001,1111
270 DATA 1111,1001,1111,0011,0011
280 !display time
290 INPUT "ENTER TIME (HH:MM) ":TIME$ :: CALL CLEAR
300 X$=TIME$ :: GOSUB 2000
310 !increment time
320 H2=VAL(SEG$(TIME$,1,1)):: H1=VAL(SEG$(TIME$,2,1)):: S2=VAL(SEG$(TIME$,4,1)):
: S1=VAL(SEG$(TIME$,5,1))
330 S1=S1+1
340 IF S1>9 THEN 400
350 IF S2>5 THEN 410
360 IF H1>9 THEN 420
370 IF H2=1 AND H1>2 THEN 430
380 TIME$=STR$(H2)&STR$(H1)&SEG$(TIME$,3,1)&STR$(S2)&STR$(S1):: GOTO 440
400 S1=0 :: S2=S2+1 :: GOTO 350
410 S2=0 :: H1=H1+1 :: GOTO 360
420 H1=0 :: H2=H2+1 :: TOTO 370
430 H2=0 :: H1=1 :: GOTO 380
440 Y$=X$ :: X$=TIME$ :: GOTO 300
1000 !display large digit D at X,Y
1010 FOR I8=0 TO 4
1020 DISPLAY AT(Y+I8,X):G$(D+1,I8+1);
1040 NEXT I8
1050 RETURN
2000 !digitial clock subroutine
2010 J=0
2020 FOR I=1 TO 5
2030 IF I=3 THEN 2060
2040 J=J+1
2050 T(J)=VAL(SEG$(X$,I,1))
2060 NEXT I
2070 J=1 :: Y=10
2080 FOR I=1 TO 5
2090 IF I=3 THEN DISPLAY AT(11,13):CHR$(30);:: DISPLAY AT(13,13):CHR$(30);:: GOT
O 2120
2100 D=T(J):: X=-4+5*I
2110 GOSUB 1000 :: J=J+1
2120 NEXT I
2130 !count 60 seconds
2140 FOR S=1 TO 17200 :: NEXT S
2150 RETURN
9999 END
```

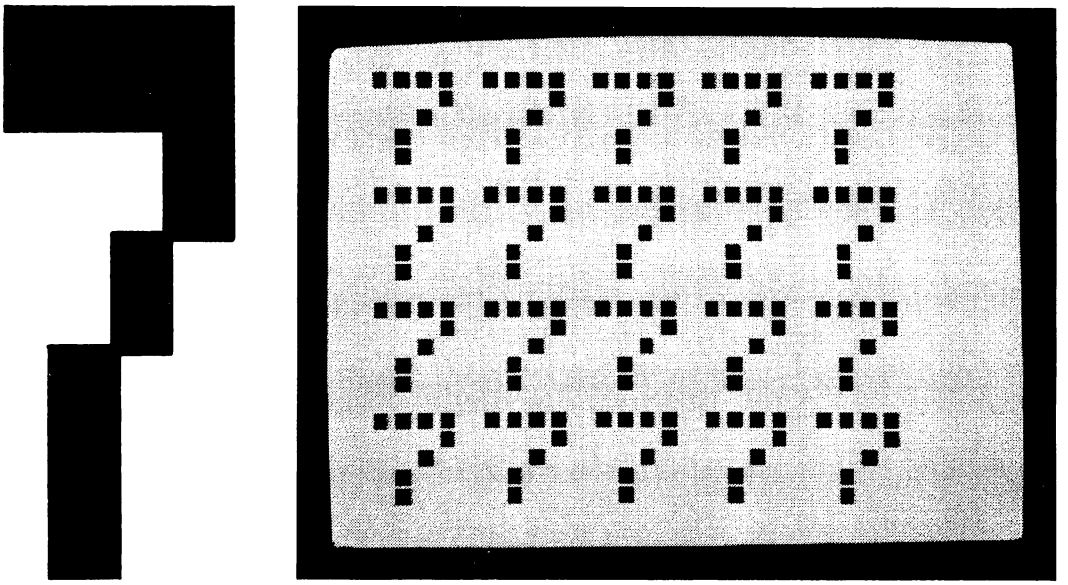


Discussion

- The oversize digit routines from program GR6P6 are used.

Suggestions

- Modify the program so that the user can specify the position of the clock on the screen.
- Alter the program so that hours, minutes, and seconds are displayed.



Coordinate Graphics

In the last chapter you saw how you could form shapes on the screen using the large blip character CHR$(30). Often it is desirable to consider the screen as a rectangular grid that can be addressed using Cartesian coordinates rather than row-column coordinates. That is, each column position corresponds to an X-position and each row position corresponds to a Y-position in the Cartesian plane. If you think of the screen this way, you can visualize each one of the 675 addressable positions of the screen as having an X,Y address instead of a row-column address. See figure 7.1.

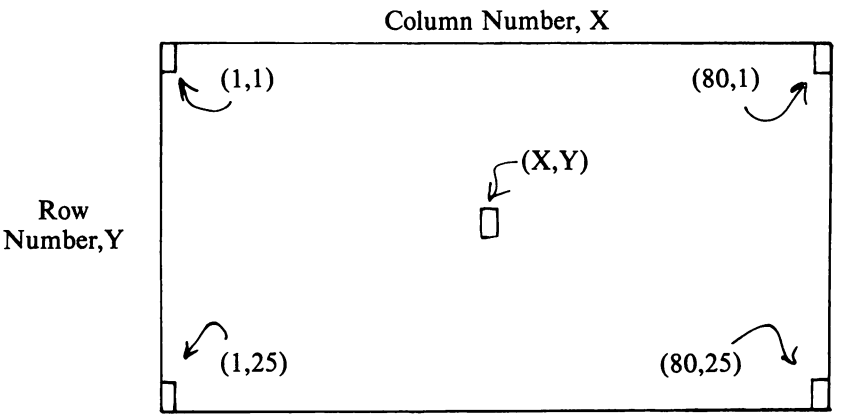


Figure 7.1 Coordinate Addressing

The DISPLAY AT instruction has two arguments, row and column, whose order is reversed from the usual way you think about Cartesian coordinates. That is, if X = 5 and Y = 10, you can visualize the 5th position over from the left and the 10th position down. You can reach this location with the instruction DISPLAY AT 10,5 which places the cursor at the Cartesian coordinates (5,10).

Consider the examples below, using A$ = CHR$(30):

Coordinates	Instruction	Output
(1,1)	`10 DISPLAY AT(1,1):A$`	Blip at upper left
(2,4)	`20 DISPLAY AT(4,2):A$`	Blip on 4th row, 2nd col
(27,24)	`30 DISPLAY AT(24,27):A$`	Blip at bottom right
(1,24)	`40 DISPLAY AT(24,1):A$`	Blip at bottom left
(27,1)	`50 DISPLAY AT(1,27):A$`	Blip at upper right

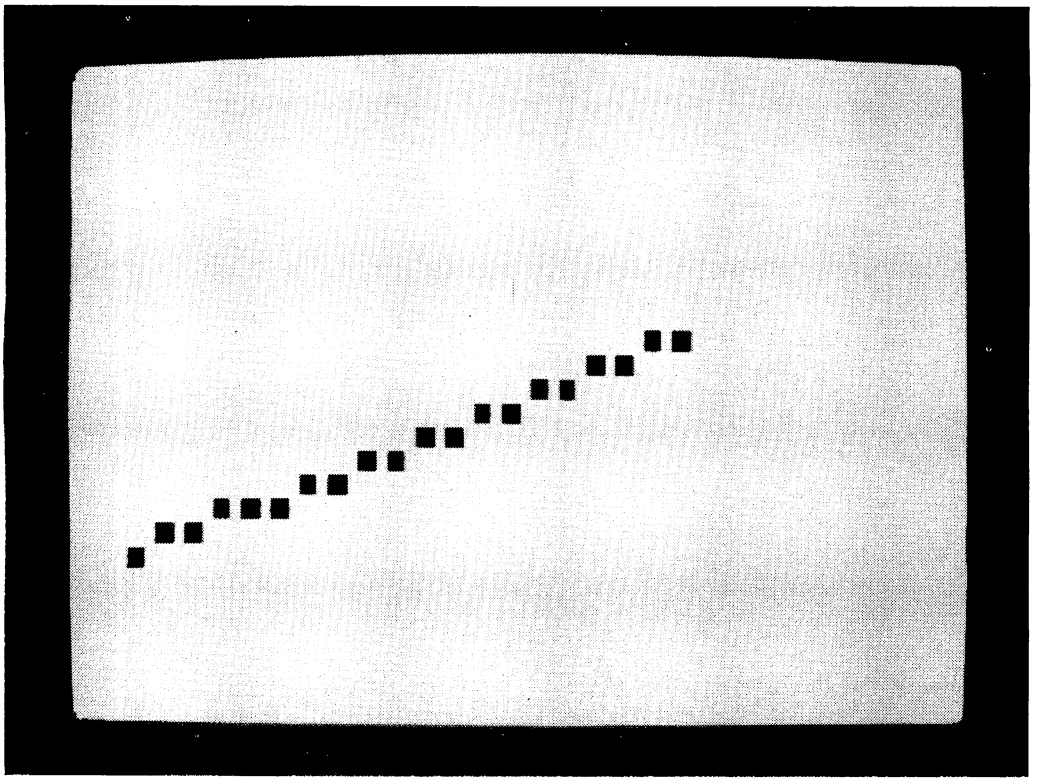
Therefore you must transform the order of the X,Y coordinates when you use the DISPLAY AT instruction and write them as DISPLAY AT(Y,X)

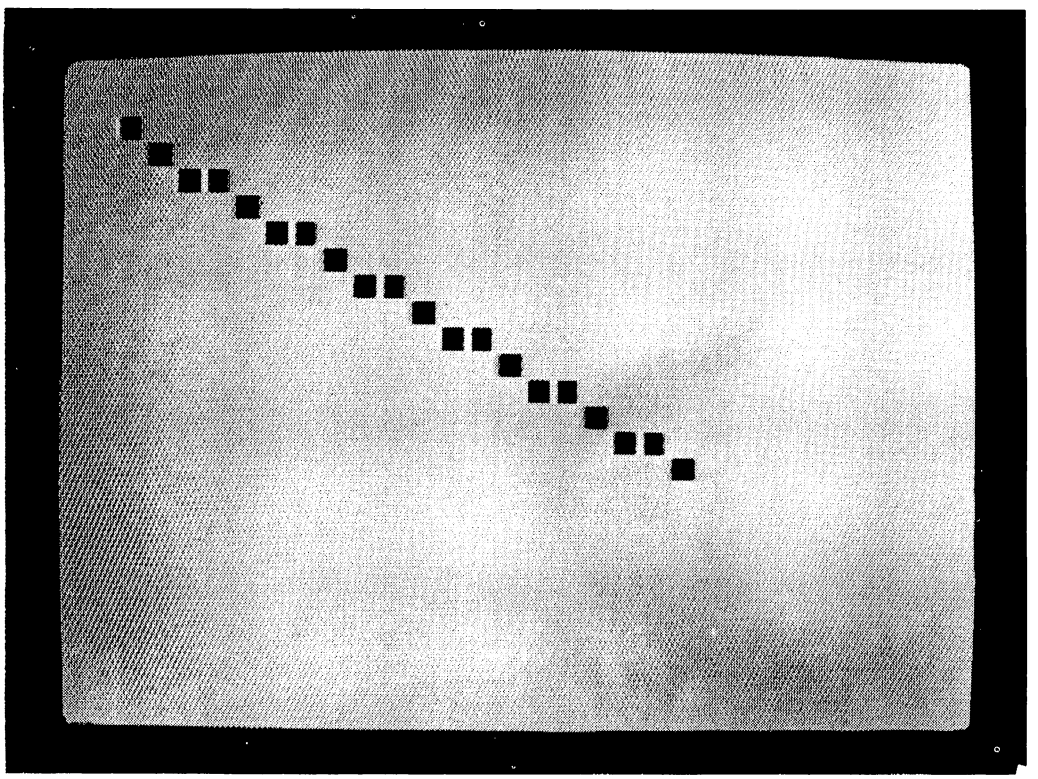
Also, you must remember that in the usual Cartesian coordinate plane, an increase in the Y value indicates the UP direction on the plane. But with the TI–99/4A an increase in the Y value indicates the DOWN direction on the screen.

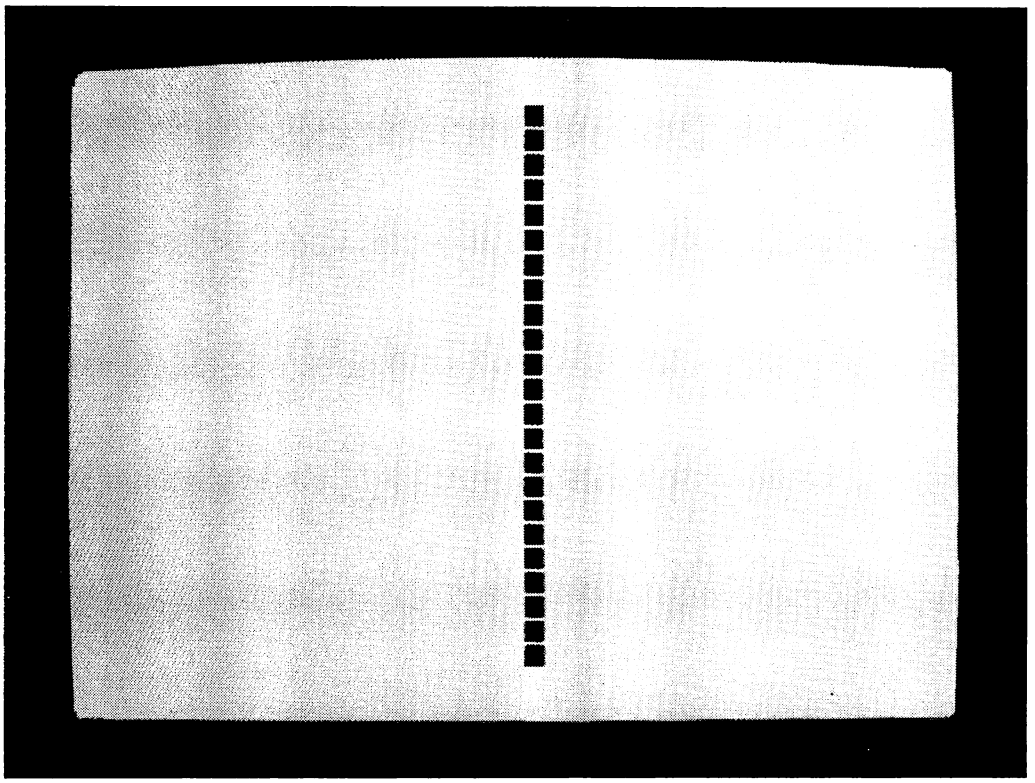
Problem 7.1 Write a program to draw a line using coordinate graphics. The user is to input the X,Y coordinates of the two end points.

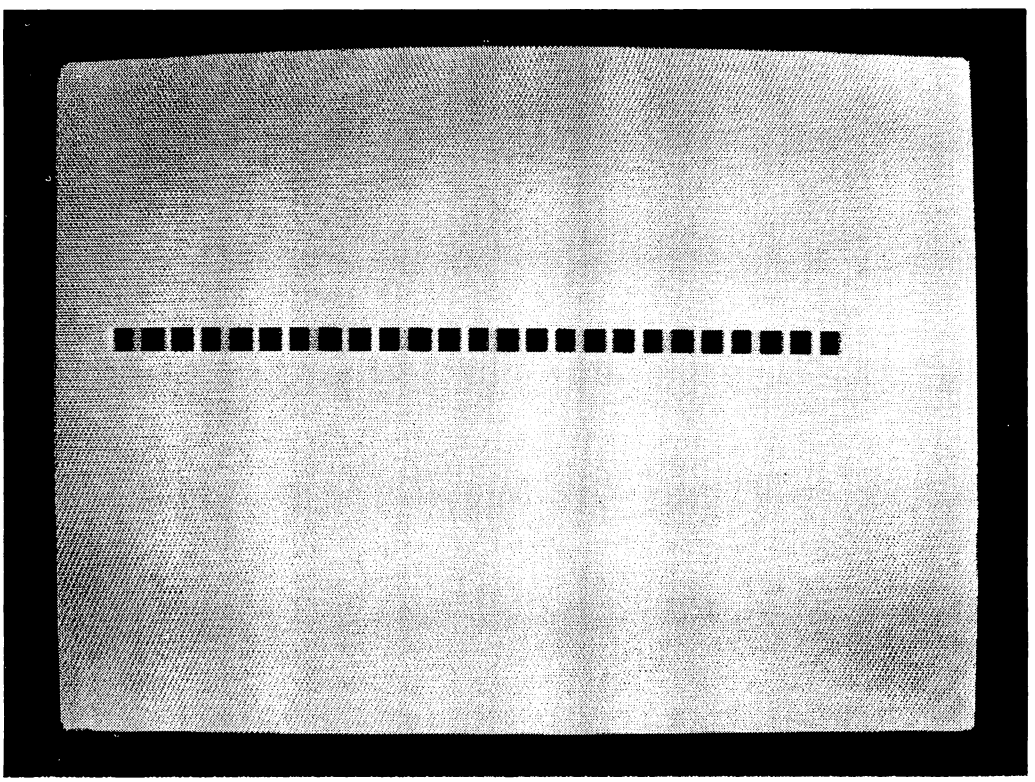
Solution

```
10 ! filename:"gr7p1"
20 !  purpose: draw a straight line given two end points
30 !   author: jpg, jdr & tfz 10/83
40 !
50 INPUT "input X,Y coordinates of end point 1 ":X1,Y1
60 INPUT "input X,Y coordinates of end point 2 ":X2,Y2
70 IF 100*X1+Y1>100*X2+Y2 THEN T=X1 :: X1=X2 :: X2=T :: Y1=Y2 :: Y2=T
80 GOSUB 1000 :: !draw the line
100 CALL KEY(0,N,S):: IF S=0 THEN 100 ELSE 9999
1000 !subroutine to draw picture on screen
1010 CALL CLEAR
1020 IF X1>23 OR X2>23 OR Y1>27 OR Y2>27 THEN PRINT "picture will not fit, canno
t continue" :: RETURN
1030 IF X1=X2 THEN 1190
1040 IF Y1=Y2 THEN 1230
1050 M=(Y2-Y1)/(X2-X1):: !calculate slope of line
1060 B=Y1-M*X1 :: !calculate y-intercept
1070 IF M>0 THEN IF M<=1 THEN 1140 ELSE 1090 ELSE IF M>=-1 THEN 1140 ELSE 1080
1080 T=Y1 :: Y1=Y2 :: Y2=T :: !slope between -1 and -infinity
1090 FOR J=Y1 TO Y2 :: !slope between 1 and infinity
1100 I=INT((J-B)/M+.5)
1110 DISPLAY AT(J,I):CHR$(30);
1120 NEXT J
1130 GOTO 1260
1140 FOR I=XI TO X2 :: !slope between -1 and 1
1150 J=INT(M*I+B+.5)
1160 DISPLAY AT(J,I):CHR$(30);
1170 NEXT I
1180 GOTO 1260
1190 FOR J=Y1 TO Y2 :: !no slope, vertical line
1200 DISPLAY AT(J,X1):CHR$(30);
1210 NEXT J
1220 GOTO 1260
1230 FOR I=X1 TO X2 :: !zero slope, horizontal line
1240 DISPLAY AT(Y1,I):CHR$(30);
1250 NEXT I
1260 RETURN
9999 END
```









Discussion

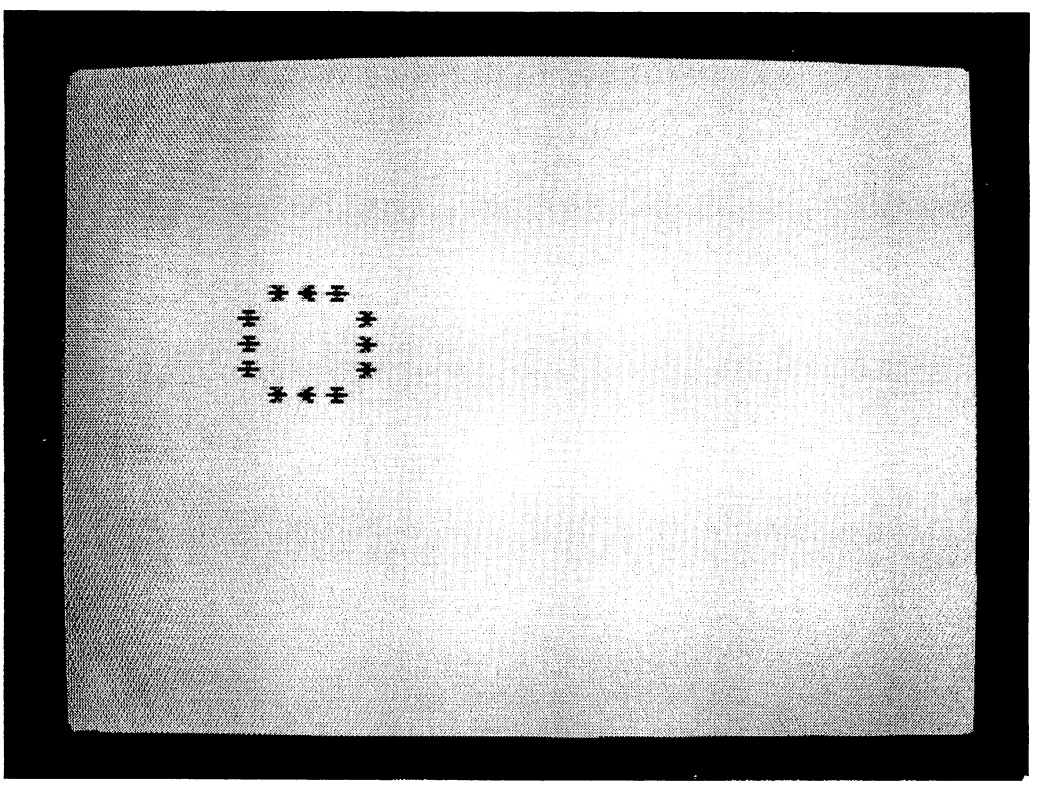
- The subroutine used to draw the line is basically the same as that in program GR5P4 with all DISPLAY AT statements replaced with X,Y coordinate equivalent commands.
- The two points are ordered in line 70 so that the drawing of the line can proceed from left to right on the screen.

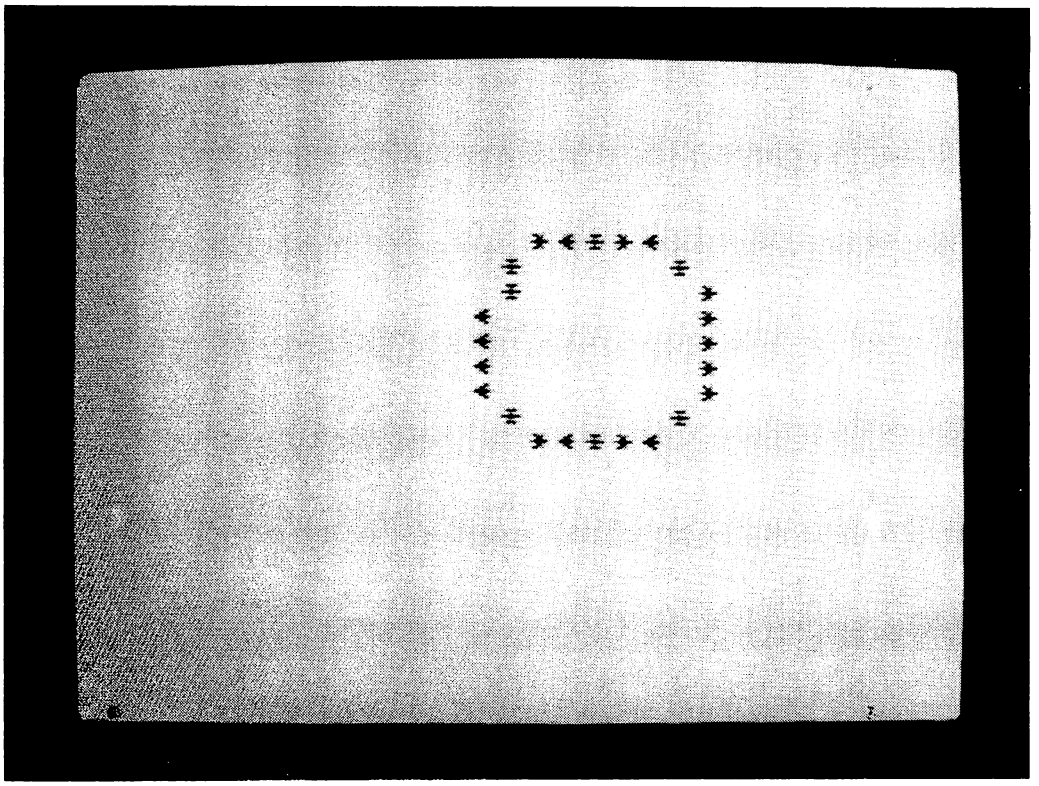
Problem 7.2

Draw a circle centered at a point X,Y with radius R.

Solution

```
10 ! filename:"gr7p2"
20 ! purpose: draw a circle
30 !   author: jpg, jdr & tfz 10/83
40 !
50 INPUT "input X,Y coordinates of circle's center ":X1,Y1
60 INPUT "input radius of circle ":R
70 CALL CLEAR :: E=2.126
80 FOR T=0 TO 6.3 STEP 1/(R+R)
90 X=INT(E*R*COS(T)+X1+.5)
100 Y=INT(E*R*SIN(T)+Y1+.5)
110 DISPLAY AT(Y,X):CHR$(42);
120 NEXT T
140 END
```





Discussion

- The polar coordinate form for the equation of a circle is used.
- Since the blips are not square, the variable E was used to help overcome inherent distortion.

Suggestions

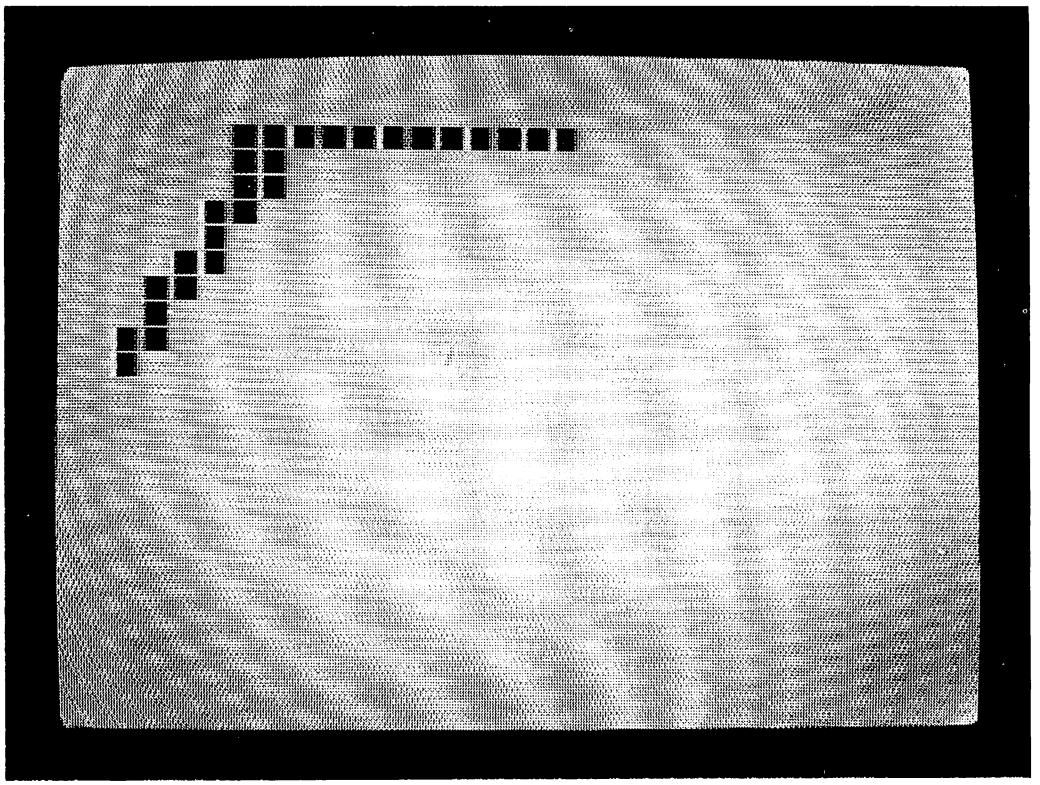
- Modify the program so that there is a multiplier variable in front of R*SIN(T) as well as R*COS(T). By varying these values, ellipses can be drawn.
- Write a program that draws circles at randomly chosen points on the screen. The radius of each circle can be selected at random also.

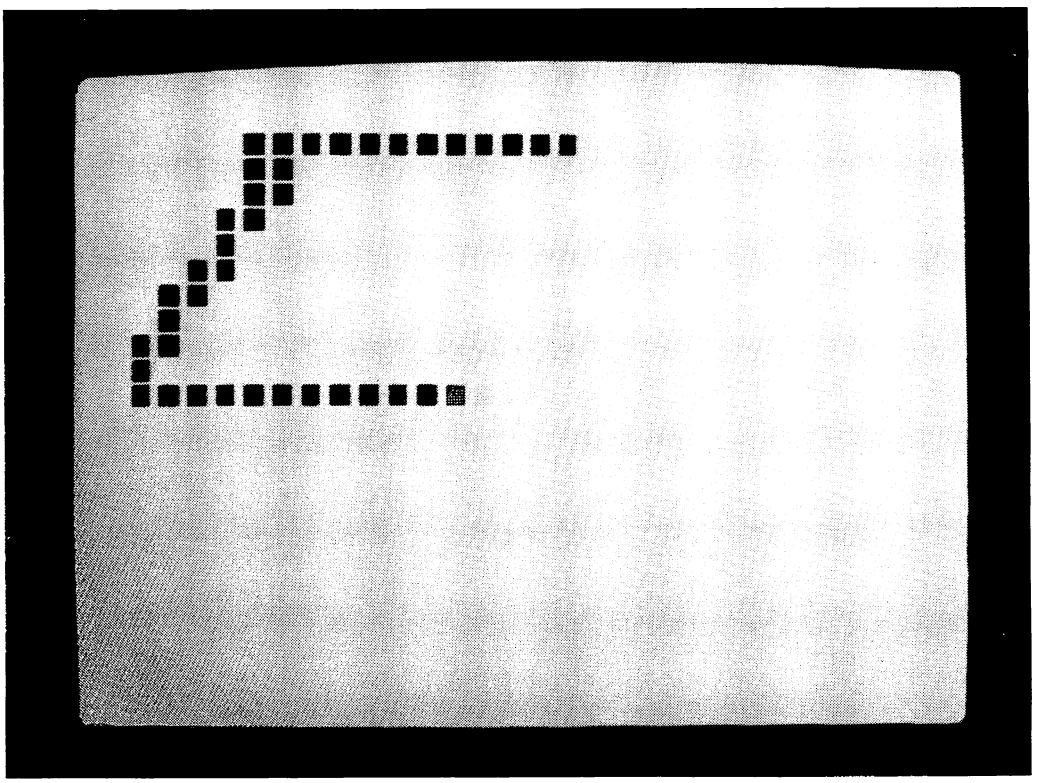
Problem 7.3

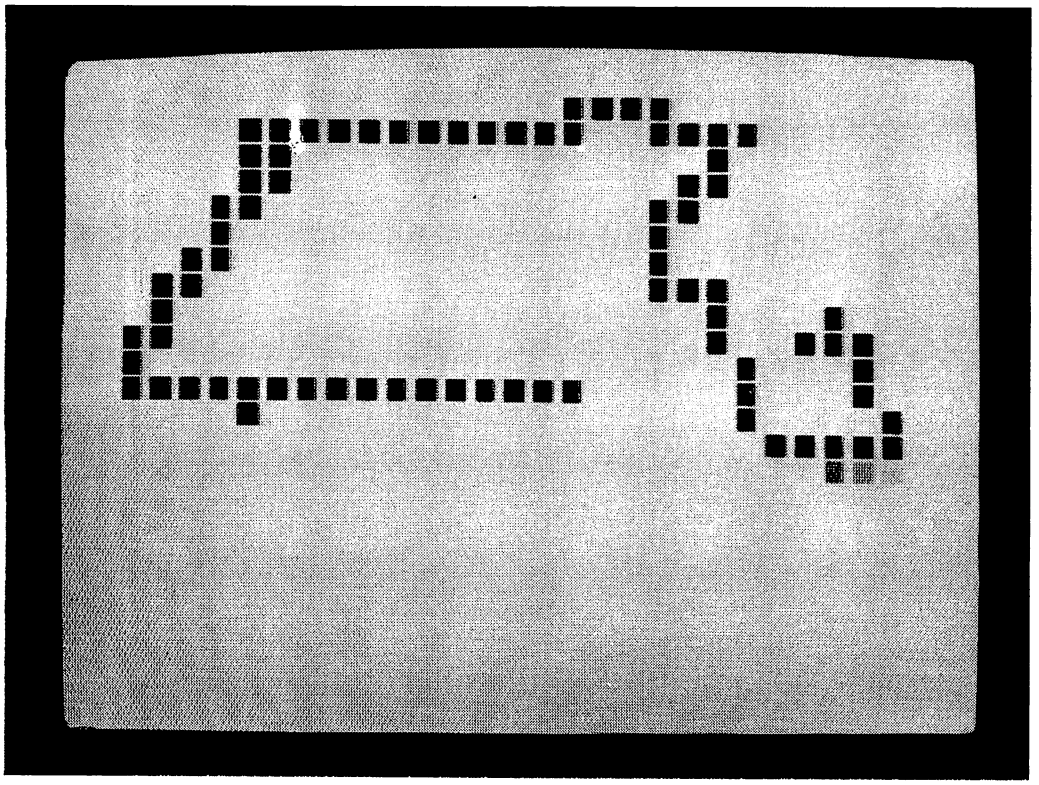
Draw an outline of the state of Massachusetts using coordinate graphics.

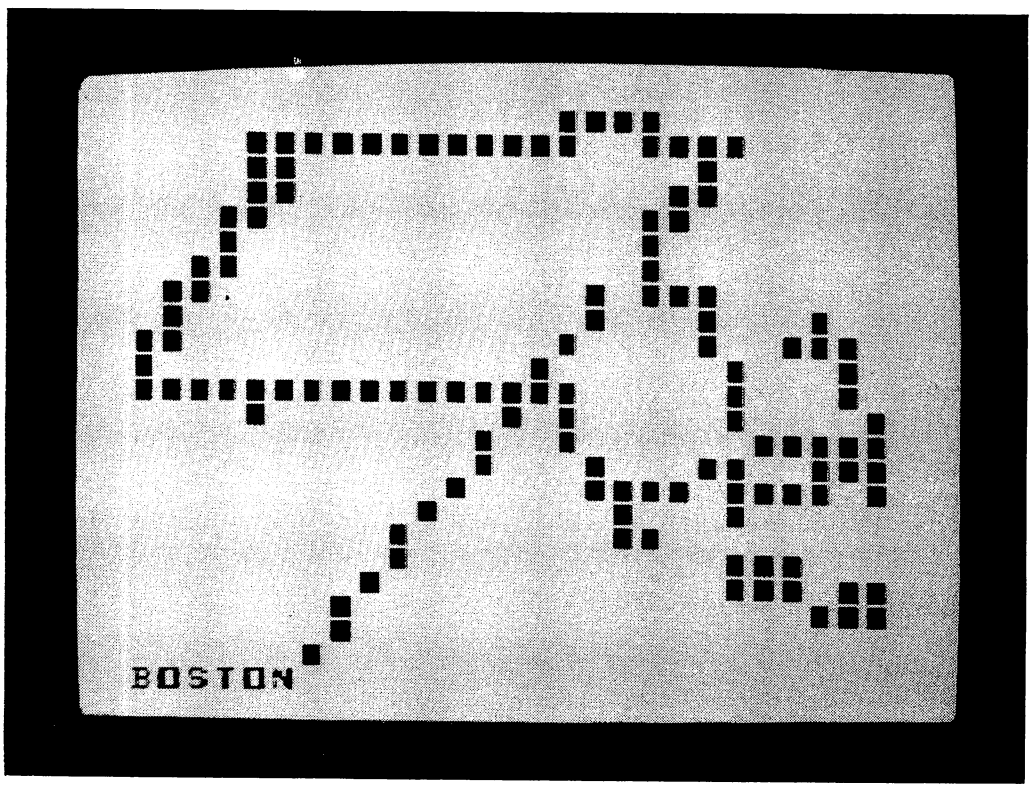
Solution

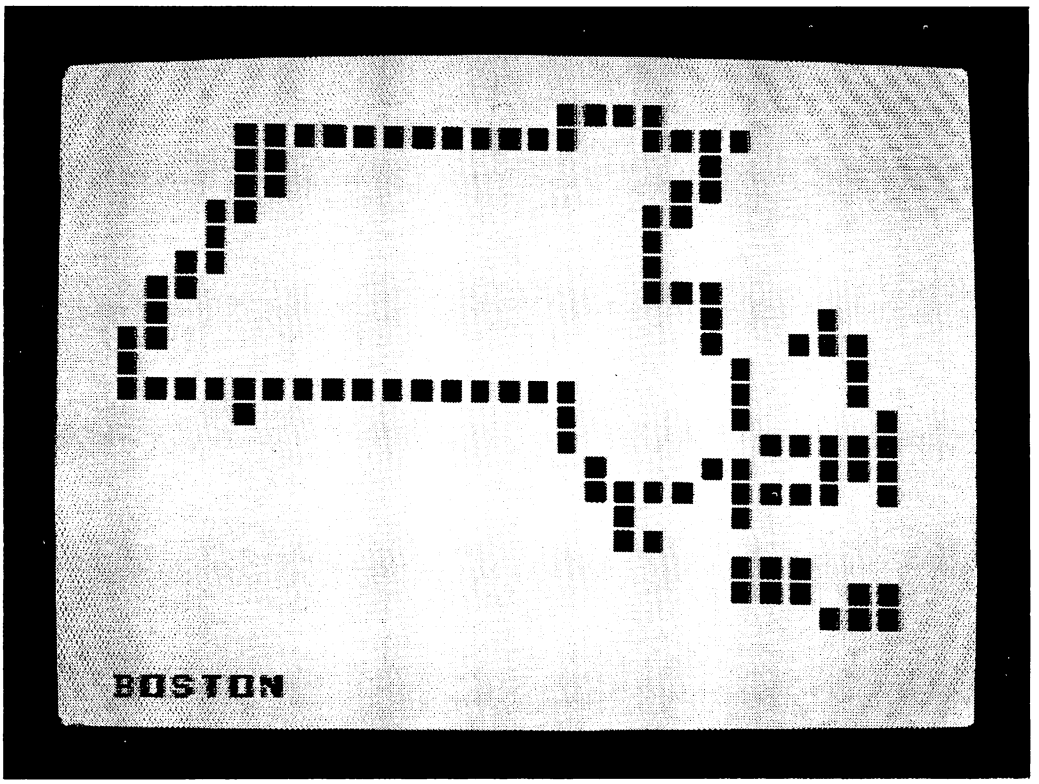
```
10 ! filename:"gr7p3"
20 !  purpose: draw picture of massachusetts
30 !   author: jpg, jdr & tfz 10/83
40 !
50 XINC=0 :: YINC=0 :: A$="BOSTON" :: X=18 :: Y=7 :: S=0
60 GOSUB 1000
70 GOSUB 6000
80 RESTORE
90 CALL KEY(0,N,S):: IF S=0 THEN 90 ELSE 50
1000 !subroutine to draw picture on screen
1010 CALL CLEAR
1020 READ N :: !number of straight lines to draw in picture
1030 FOR L=1 TO N
1040 READ C,D
1050 X1=INT(C/100):: Y1=C-100*X1
1060 X2=INT(D/100):: Y2=D-100*X2
1070 X1=X1+XINC :: Y1=Y1+YINC :: !relocate coordinates
1080 X2=X2+XINC :: Y2=Y2+YINC
1100 IF X1=X2 THEN 1260
1110 IF Y1=Y2 THEN 1300
1120 M=(Y2-Y1)/(X2-X1):: !calculate slope of line
1130 B=Y1-M*X1 :: !calculate y-intercept
1140 IF M>0 THEN IF M<=1 THEN 1210 ELSE 1160 ELSE IF M>=-1 THEN 1210 ELSE 1150
1150 T=Y1 :: Y1=Y2 :: Y2=T :: !slope between -1 and -infinity
1160 FOR J=Y1 TO Y2 :: !slope between 1 and infinity
1170 I=INT((J-B)/M+.5)
1180 DISPLAY AT(J,I):CHR$(30);
1190 NEXT J
1200 GOTO 1330
1210 FOR I=X1 TO X2 :: !slope between -1 and 1
1220 J=INT(M*I+B+.5)
1230 DISPLAY AT(J,INT(I*.35)):CHR$(30);
1240 NEXT I
1250 GOTO 1330
1260 FOR J=Y1 TO Y2 :: !no slope, vertical line
1270 DISPLAY AT(J,INT(X1*.35)):CHR$(30);
1280 NEXT J
1290 GOTO 1330
1300 FOR I=X1 TO X2 :: !zero slope, horizontal line
1310 DISPLAY AT(Y1,INT(I*.35)):CHR$(30);
1320 NEXT I
1330 IF S=1 THEN RETURN
1340 NEXT L
1350 RESTORE :: S=1
1360 RETURN
2000 DATA 42,110,602,111,702,1602,4602,112,4712,1613,1713
2010 DATA 4701,4801,4901,5601,5701,5702,6203,6203,5802,6302
2020 DATA 5904,6104,5605,5905,5506,5507,5608,6008,6009,6110
2030 DATA 6210,6412,6513,6614,6714,7514,7110,7110,7209,7409
2040 DATA 7310,7510,7511,7511,7611,7914,7415,7815,7816,7816
2050 DATA 6616,7416,6315,6517,6417,6417,6015,6315,5916,5916
2060 DATA 5616,5816,5316,5518,5116,5216,5418,5418,4915,5015
2070 DATA 4714,4814,4613,4713,6320,6920,6419,6619,6919,6919
2080 DATA 7421,7821,7720,7820
6000 !draw line from town name to position in map
6010 X1=7 :: Y1=23 :: X2=X+XINC :: Y2=Y+YINC
6020 DISPLAY AT(24,1):A$;:: DISPLAY AT(Y2,X2):CHR$(30);
6030 GOSUB 1120
6040 X2=X+XINC :: Y2=Y+YINC
6050 FOR J=1 TO 20
6060 FOR I=1 TO 100 :: NEXT I :: DISPLAY AT(Y2,X2):" ";
6070 FOR I=1 TO 100 :: NEXT I :: DISPLAY AT(Y2,X2):CHR$(30);
6080 NEXT J
6090 RETURN
9999 END
```








BOSTON



Discussion

- Again, the basic line drawing subroutine that has been used in previous programs appears at lines 1000 and beyond with DISPLAY AT and PRINT commands doing the drawing on the screen.
- The DATA statements provide end points of the lines that are drawn to form the picture.
- As an extra flourish, the town of Boston is singled out for special identification.
- Finally with panache, the position of the town blinks on and off.

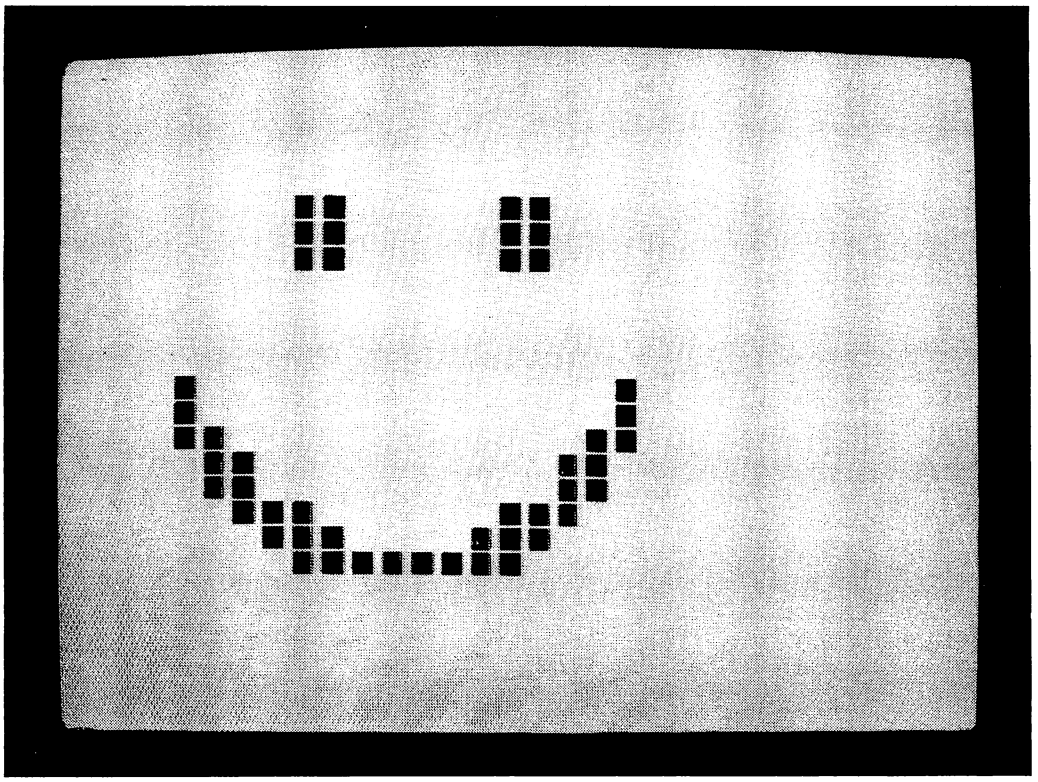
Suggestions

- Modify the program to give the user a menu of cities to choose from, with the program then drawing a line from the town's name to its position in the state.

Problem 7.4 Write a program that will produce a smile face at a position on the screen specified by the user.

Solution

```
10 ! filename:"gr7p4"
20 !  purpose: smile face w/ mobility
30 !   author: jpg, jdr & tfz 10/83
40 !
50 CALL CLEAR
60 INPUT "input position of picture ":XINC,YINC
70 GOSUB 1000
80 RESTORE
90 CALL KEY(0,N,S):: IF S=0 THEN 90 ELSE 50
1000 !subroutine to draw picture on screen
1010 CALL CLEAR
1020 READ N :: !number of straight lines to draw in picture
1030 FOR L=1 TO N
1040 READ C,D
1050 X1=INT(C/100):: Y1=C-100*X1
1060 X2=INT(D/100):: Y2=D-100*X2
1070 X1=X1+XINC :: Y1=Y1+YINC :: !relocate coordinates
1080 X2=X2+XINC :: Y2=Y2+YINC
1090 IF INT(X1*.35)>27 OR INT(X2*.35)>27 OR Y1>24 OR Y2>24 THEN PRINT "picture w
ill not fit, cannot continue" :: RETURN
1100 IF X1=X2 THEN 1260
1110 IF Y1=Y2 THEN 1300
1120 M=(Y2-Y1)/(X2-X1):: !calculate slope of line
1130 B=Y1-M*X1 :: !calculate y-intercept
1140 IF M>0 THEN IF M<=1 THEN 1210 ELSE 1160 ELSE IF M>=-1 THEN 1210 ELSE 1150
1150 T=Y1 :: Y1=Y2 :: Y2=T :: !slope between -1 and -infinity
1160 FOR J=Y1 TO Y2 :: !slope between 1 and infinity
1170 I=INT((J-B)/M+.5)
1180 DISPLAY AT(J,I):CHR$(30);
1190 NEXT J
1200 GOTO 1330
1210 FOR I=X1 TO X2 :: !slope between -1 and 1
1220 J=INT(M*I+B+.5)
1230 DISPLAY AT(J,INT(I*.35)):CHR$(30);
1240 NEXT I
1250 GOTO 1330
1260 FOR J=Y1 TO Y2 :: !no slope, vertical line
1270 DISPLAY AT(J,INT(X1*.35)):CHR$(30);
1280 NEXT J
1290 GOTO 1330
1300 FOR I=X1 TO X2 :: !zero slope, horizontal line
1310 DISPLAY AT(Y1,INT(I*.35)):CHR$(30);
1320 NEXT I
1330 NEXT L
1340 RETURN
2000 DATA 25,1200,1202,1300,1302,3000,3002,3400,3402,7,9
2010 DATA 107,109,4207,4209,4307,4309,209,210,309,310,4009,4010
2020 DATA 4109,4110,410,411,510,511,3810,3811,3910,3911,611,611
2030 DATA 711,711,3611,3611,3711,3711,612,1112,3212,3712
2040 DATA 813,1513,2813,3513,1214,3114
9999 END
```



Discussion

- Notice the similarity between this and the previous program. The generality of this approach for producing graphics is quite astonishing.

Suggestions

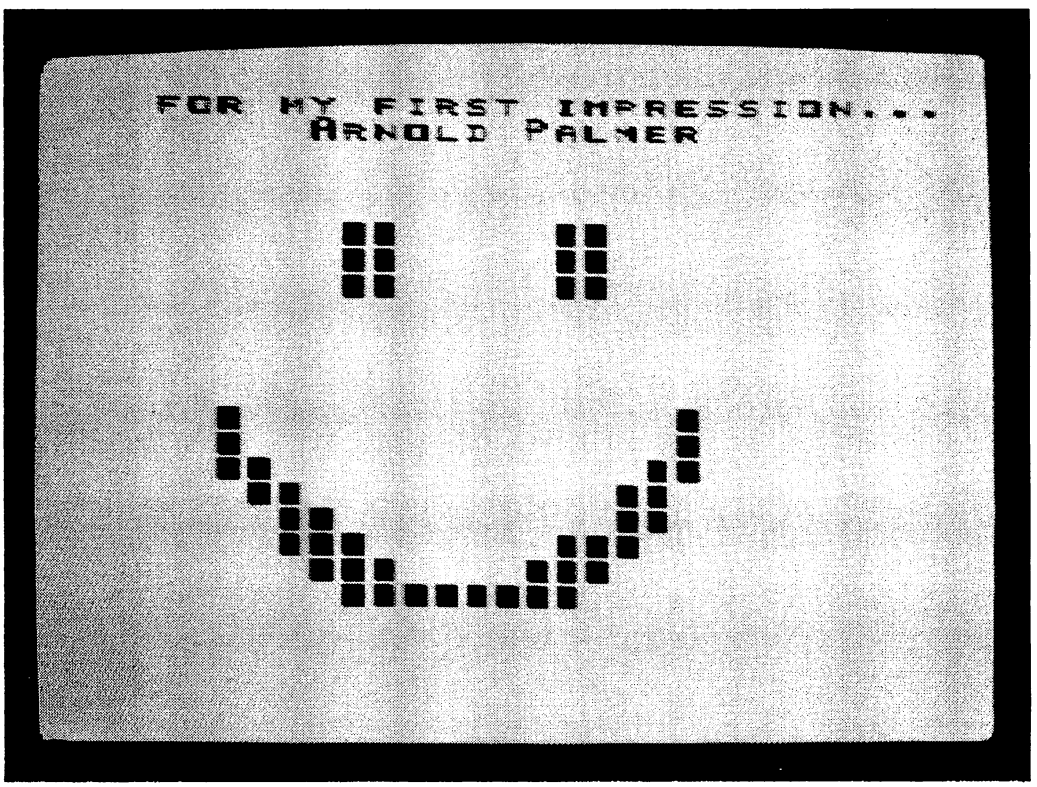
- Modify the program so that a circle delineating the head is drawn around the eyes and smile.

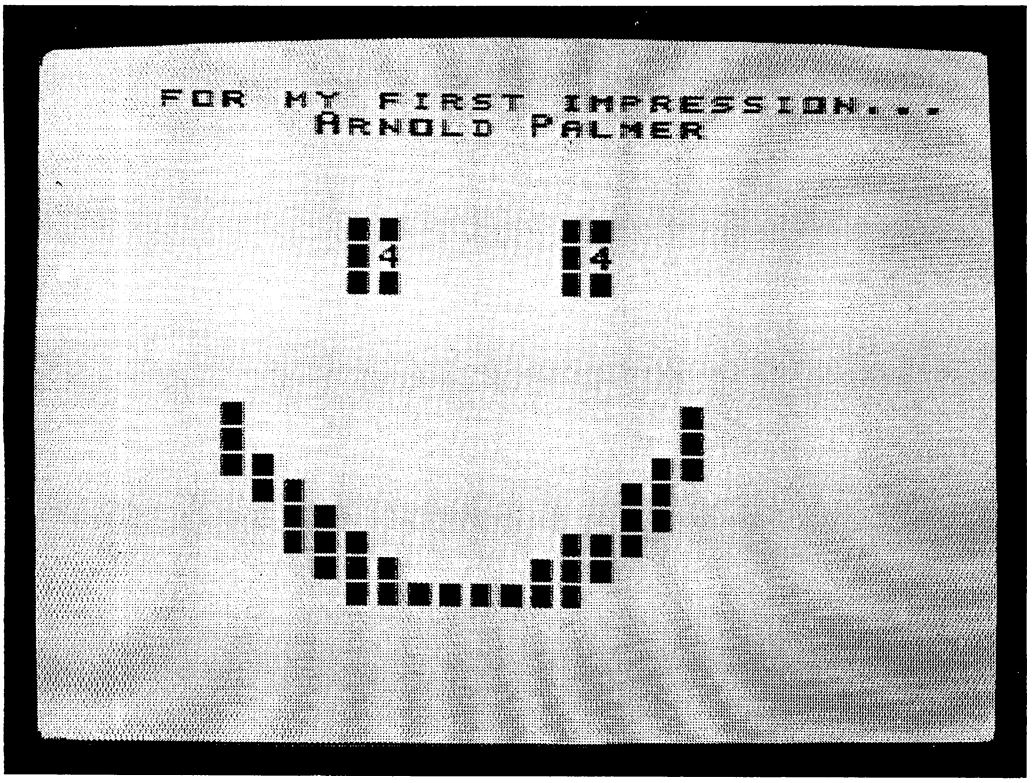
Problem 7.5 Write a program that turns the computer into a stand up comedian who does impressions.

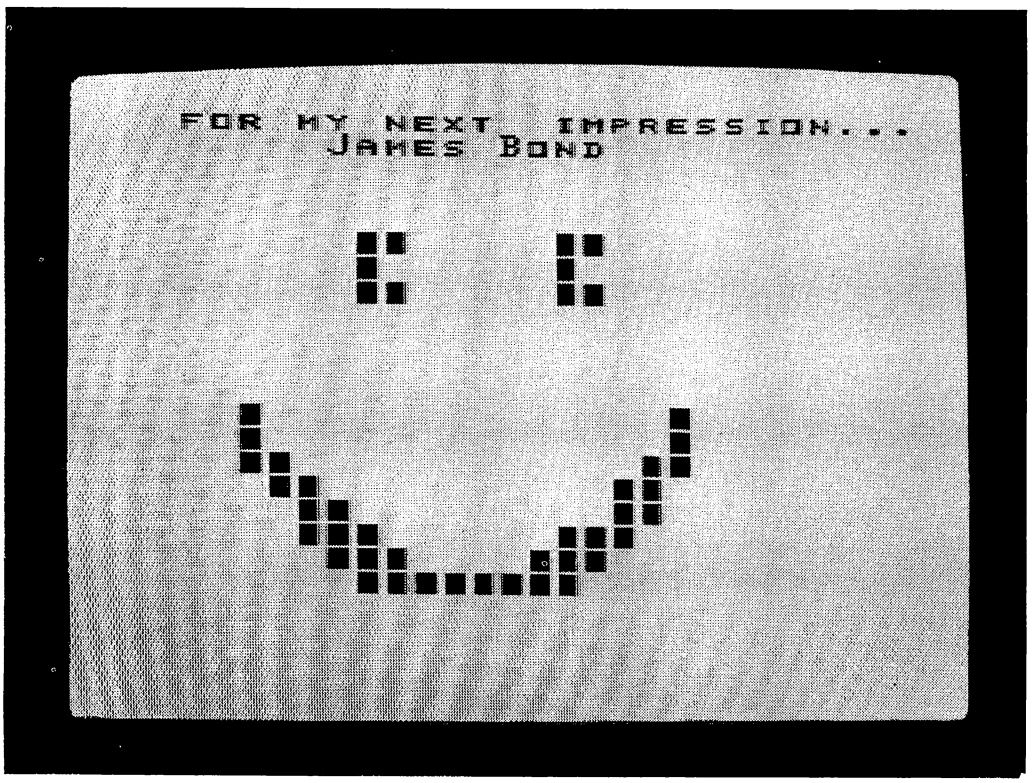
Solution

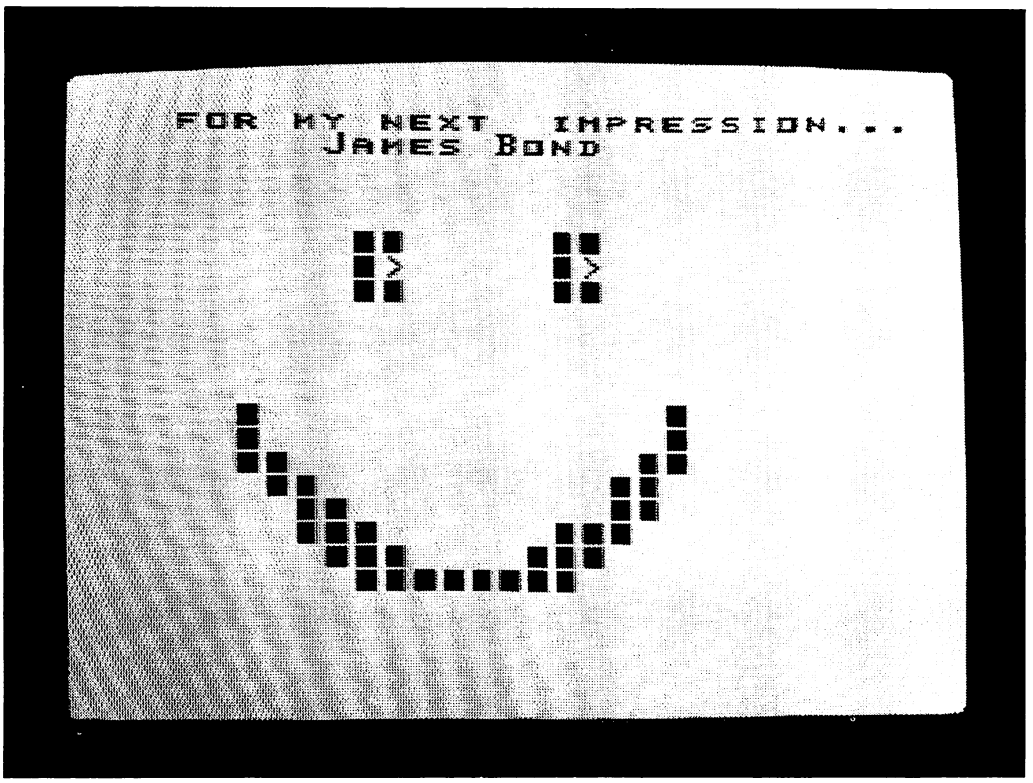
```
10 ! filename:"gr7p5"
20 !  purpose: stand up comedian doing impersonations
30 !   author: jpg, jdr & tfz 10/83
40 !
50 DIM W$(14),C$(14)
60 FOR I=1 TO 14 :: READ W$(I),A,B,C
70 C$(I)=CHR$(A)&CHR$(B)&CHR$(C)
80 NEXT I
90 CALL CLEAR
100 XINC=16 :: YINC=6
110 DISPLAY AT(1,2):"And now,";:: DISPLAY AT(2,4):" heeeeeres John Binary..."
120 FOR I=1 TO 999 :: NEXT I
130 GOSUB 1000 :: !draw smile face
140 FOR Q=1 TO 14 :: DISPLAY AT(1,3):"for my ";
160 IF Q=1 THEN DISPLAY AT(1,10):"first";ELSE IF Q=14 THEN DISPLAY AT(1,10):"las
t";ELSE DISPLAY AT(1,10):"next ";
170 DISPLAY AT(1,16):"impression...";
180 DISPLAY AT(2,8):W$(Q);
190 FOR I=1 TO 800 :: NEXT I
200 A$=C$(Q)
210 A$=A$&CHR$(30)
220 GOSUB 3000 :: !impersonation subroutine
230 DISPLAY AT(2,8):RPT$(CHR$(32),16)
240 FOR I=1 TO 600 :: NEXT I
250 NEXT Q
260 FOR I=1 TO 800 :: NEXT I
270 DISPLAY AT(24,15):"thank you..."
280 FOR I=1 TO 2200+INT(1000*RND):: NEXT I
290 DISPLAY AT(7,10):" ";
300 DISPLAY AT(7,17):" ";
310 FOR I=1 TO 100 :: NEXT I
320 DISPLAY AT(7,10):CHR$(30);
330 DISPLAY AT(7,17):CHR$(30);
340 CALL KEY(0,N,S):: IF S=0 THEN 340
350 CALL CLEAR
360 STOP
1000 !subroutine to draw picture on screen
1010 CALL CLEAR
1020 READ N :: !number of straight lines to draw in picture
1030 FOR L=1 TO N
1040 READ C,D
1050 X1=INT(C/100):: Y1=C-100*X1
1060 X2=INT(D/100):: Y2=D-100*X2
1070 X1=X1+XINC :: Y1=Y1+YINC :: !relocate coordinates
1080 X2=X2+XINC :: Y2=Y2+YINC
1090 IF INT(X1*.35)>27 OR INT(X2*.35)>27 OR Y1>24 OR Y2>24 THEN PRINT "picture w
ill not fit, cannot continue" :: RETURN
1100 IF X1=X2 THEN 1260
1110 IF Y1=Y2 THEN 1300
1120 M=(Y2-Y1)/(X2-X1):: !calculate slope of line
1130 B=Y1-M*X1 :: !calculate slope of line
1140 IF M>0 THEN IF M<=1 THEN 1210 ELSE 1160 ELSE IF M>=-1 THEN 1210 ELSE 1150
1150 T=Y1 :: Y1=Y2 :: Y2=T :: !slope between -1 and -infinity
1160 FOR J=Y1 TO Y2 :: !slope between 1 and infinity
1170 I=INT((J-B)/M*.5)
1180 DISPLAY AT(J,I):CHR$(30);
1190 NEXT J
1200 GOTO 1330
1210 FOR I=X1 TO X2 :: !slope between -1 and 1
1220 J=INT(M*I+B+.5)
1230 DISPLAY AT(J,INT(I*.35)):CHR$(30);
```

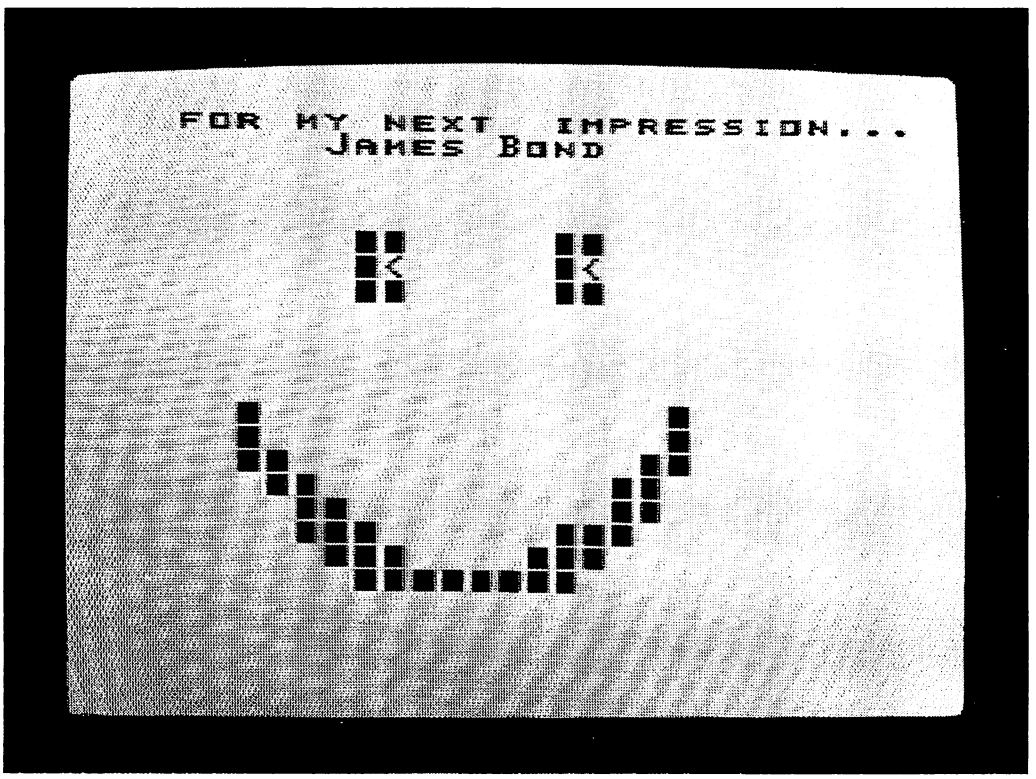
```
1240 NEXT I
1250 GOTO 1330
1260 FOR J=Y1 TO Y2 :: !no slope, vertical line
1270 DISPLAY AT(J,INT(X1*.35)):CHR$(30);
1280 NEXT J
1290 GOTO 1330
1300 FOR I=X1 TO X2 :: !zero slope, horizontal line
1310 DISPLAY AT(Y1,INT(I*.35)):CHR$(30);
1320 NEXT I
1330 NEXT L
1340 RETURN
2000 DATA "Arnold Palmer",52,52,52,"Little Orphan Annie",111,111,111
2010 DATA "James Bond",60,32,62,"Jimmy The Greek",37,37,37
2020 DATA "Rona Barrett",39,33,39,"4-year-old-child",63,32,63
2030 DATA "a math teacher",49,43,49,"a math student",61,32,50
2040 DATA "Bozo the Clown",43,120,43,"Gloria Steinem",61,32,61
2050 DATA "NBC censor",42,32,42,"Dean Martin",74,38,66
2060 DATA "Irma La Douce",36,32,36
2070 DATA "librarian in Boston's combat zone",46,111,79
2080 DATA 25,1200,1202,1300,1302,3000,3002,3400,3402,7,9
2090 DATA 107,109,4207,4209,4307,4309,209,210,309,310,4009,4010
2100 DATA 4109,4110,410,411,510,511,3810,3811,3910,3911,611,611
2110 DATA 711,711,3611,3611,3711,3711,612,1112,3212,3712
2120 DATA 813,1513,2813,3513,1214,3114
3000 !impersonation subroutine
3010 FOR I=1 TO 4 :: X$=SEG$(A$,I,1)
3020 DISPLAY AT(7,10):X$;:: DISPLAY AT(7,17):X$;
3030 FOR J=1 TO 700 :: NEXT J
3050 NEXT I
3060 RETURN
9999 END
```

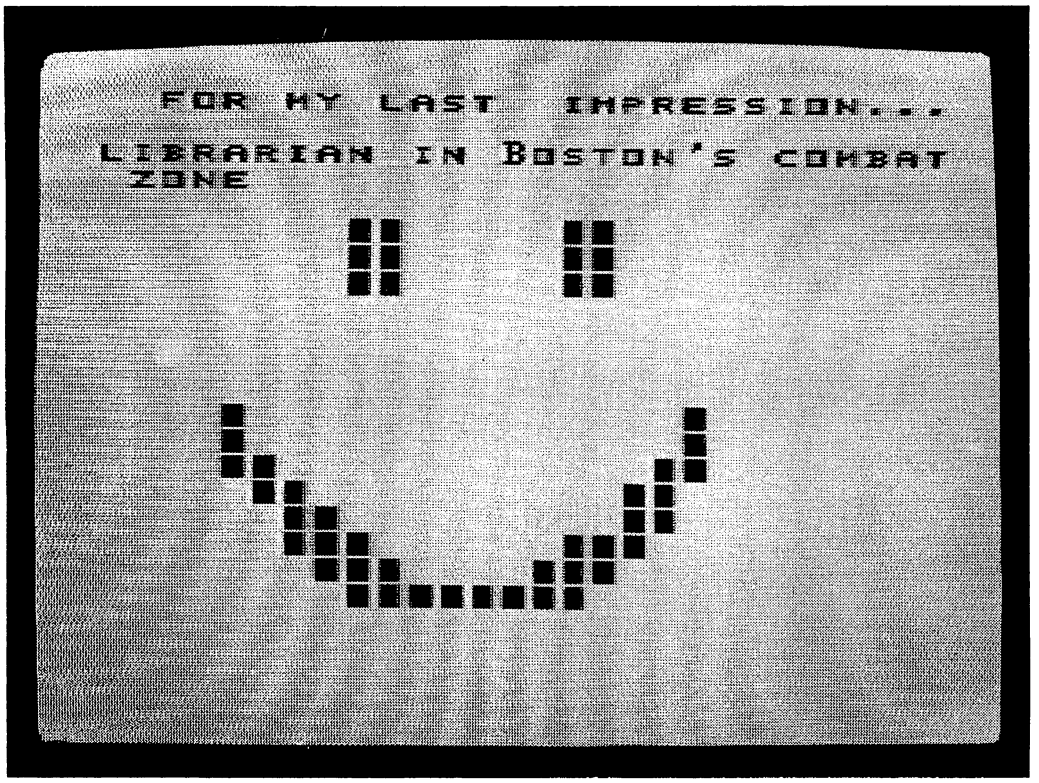



FOR MY FIRST IMPRESSION...
ARNOLD PALMER
4
4


FOR MY NEXT IMPRESSION...
JAMES BOND

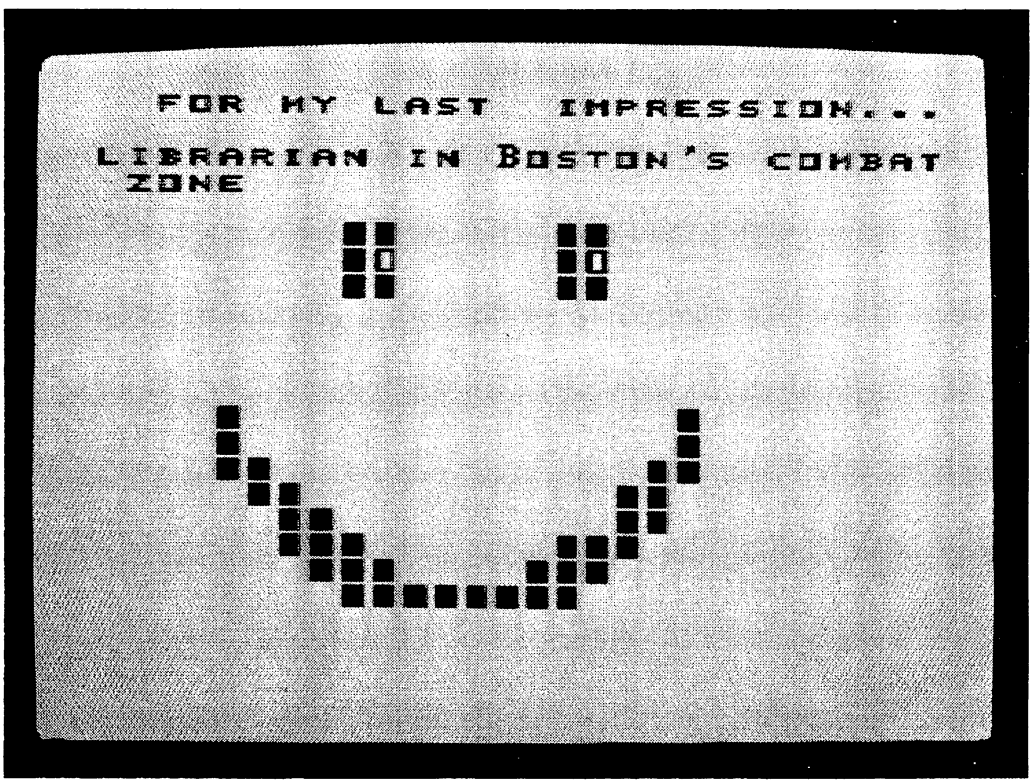

FOR MY NEXT IMPRESSION...
JAMES BOND


FOR MY NEXT IMPRESSION...
JAMES BOND


FOR MY LAST IMPRESSION...
LIBRARIAN IN BOSTON'S COMBAT ZONE


FOR MY LAST IMPRESSION...
LIBRARIAN IN BOSTON'S COMBAT ZONE


FOR MY LAST IMPRESSION...
LIBRARIAN IN BOSTON'S COMBAT ZONE

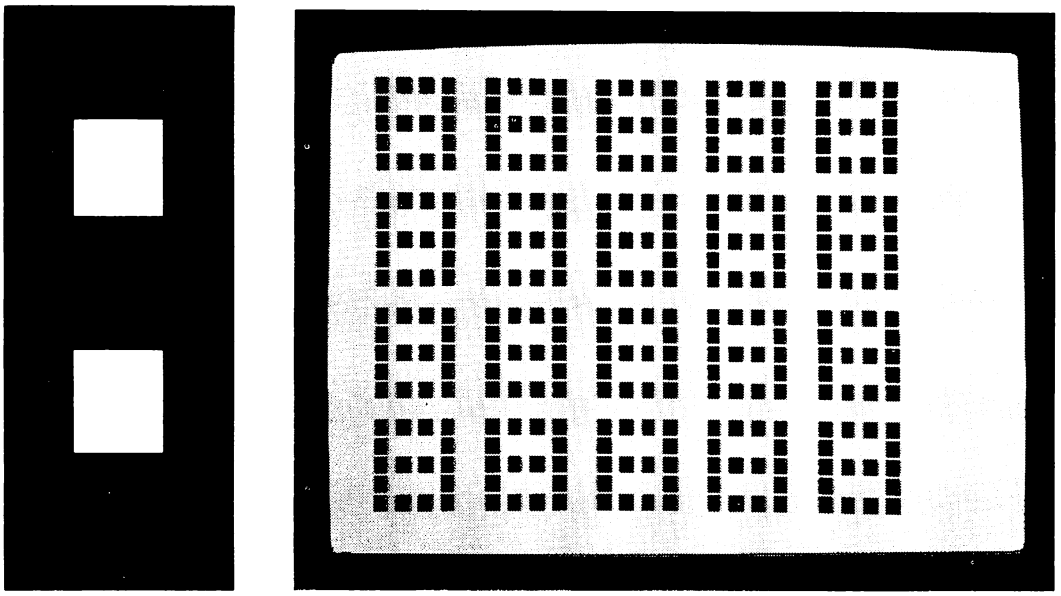

FOR MY LAST IMPRESSION...
LIBRARIAN IN BOSTON'S COMBAT ZONE

Discussion

- The program performs 14 impressions.
- Notice the simple motion that the eyes give the smile face.
- This program is a crude beginning of what might be called cartooning.
- The stand up comic's name is John Binary. Of course, his idol is John Byner.

Suggestions

- Turn the smile-faced impressionist of this program into a comedian that delivers short, snappy one-liners. His name could be Grinny Youngman. "Take my computer—please!"



Screen Store Graphics

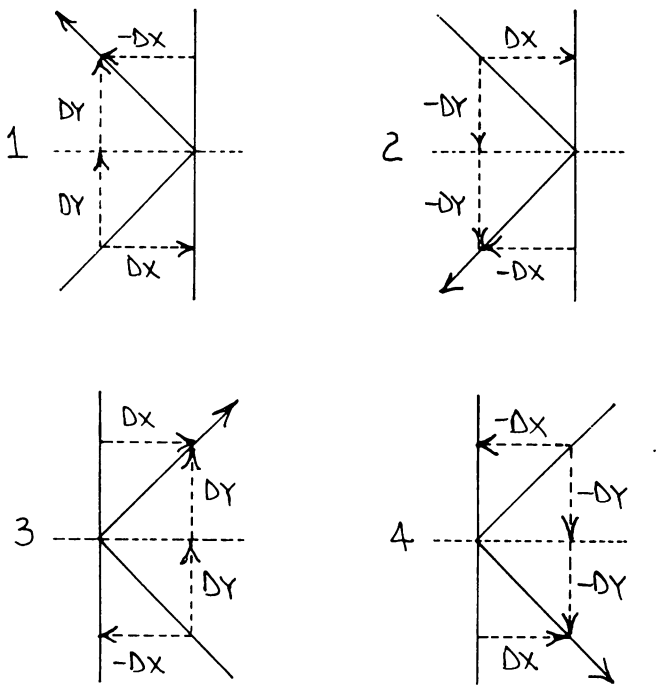
The previous chapters have shown you programs ranging widely in sophistication from a simple picture in the program Woodstock to a computerized map drawn using coordinate graphics. There was little motion in the graphics examples because it is a difficult proposition to plan on paper and realize in code the effect of movement. Many factors contribute to this difficulty. The speed of the computer is one such factor because there is usually a lot of calculation when motion is involved. The translation of where each point "moves to" requires a calculation. The resolution of the video screen is another factor governing the way a picture looks. If the picture is rotated, then distortion can occur. Much planning must take place prior to writing a program. A third factor is that the BASIC programming language was not designed to facilitate the many primitive graphics operations that would be useful. Another, and probably the most important factor impeding a computer's ability to produce motion, is a lack of imagination and willingness to experiment on the part of the computer's programmer.

We will show you some programs that will illustrate the beginnings of how motion can be produced and the visual diversity and delight it can provide.

Bouncing Dots

In many games, as well as in serious graphing applications, you will want to have a dot move in a straight direction until it meets an obstacle and keeps moving in a new direction. Consider the possible bounces:

Vertical Wall Collision



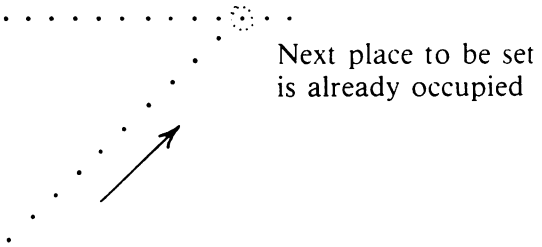
Each moving dot changes its X direction by an amount DX, and its Y direction by an amount DY. In all the conditions above, the Y direction increment DY never changes sign. In conditions 1 and 2, DX starts as positive, but changes sign to negative on the bounce.

If you investigate the four possible bounces from a horizontal wall, you will find that, as expected, the only change is the reversal of the sign of DY.

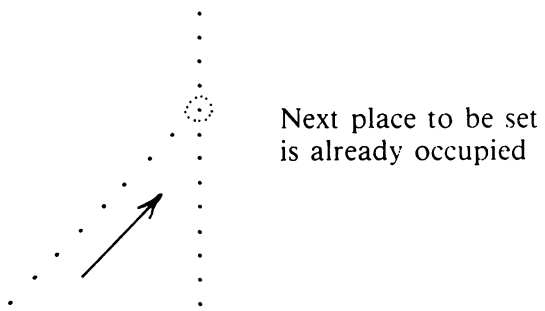
This understanding is all that is necessary to graph a moving and bouncing dot. The dot is drawn using coordinate graphics. The coordinates X,Y reverse to the equivalent DISPLAY AT (Y,X):A$ (if A$ is CHR$(30)). The movement is provided by a DISPLAY AT at a new position X + DX, Y + DY followed by a DISPLAY AT " " at the old position X,Y.

```
100 DISPLAY AT(Y,X):A$
110 X=X+DX: Y=Y+DY
120 DISPLAY AT(Y,X):A$
130 DISPLAY AT(Y-DY,X-DX):" "
140 GOTO 110
```

To bounce the dot off the wall, you must "feel" ahead to see if any part of the surrounding territory is occupied. It is not enough to just sense the status of the next point. For example, suppose the dot is moving up and to the right, and it encounters a horizontal wall.

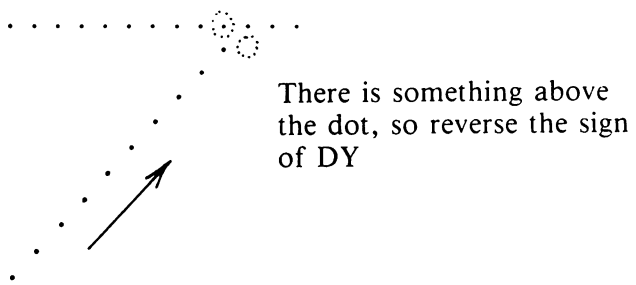


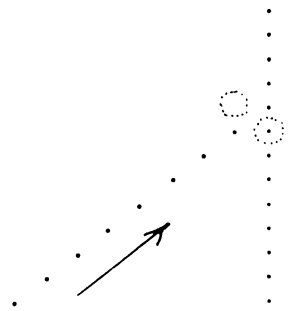
If the "next place" is the only point that is considered on a bounce, the computer couldn't tell if the wall were horizontal or vertical.



The only way to judge which direction increment, the DX or the DY, needs to change sign is to sense whether the wall is horizontal or vertical.

Examples:





There is something to the right of the dot, so reverse the sign of DX

The program that follows expands the idea above.

Problem 8.1

Write a program that bounces a dot off any wall set up in either a vertical or horizontal position.

Solution

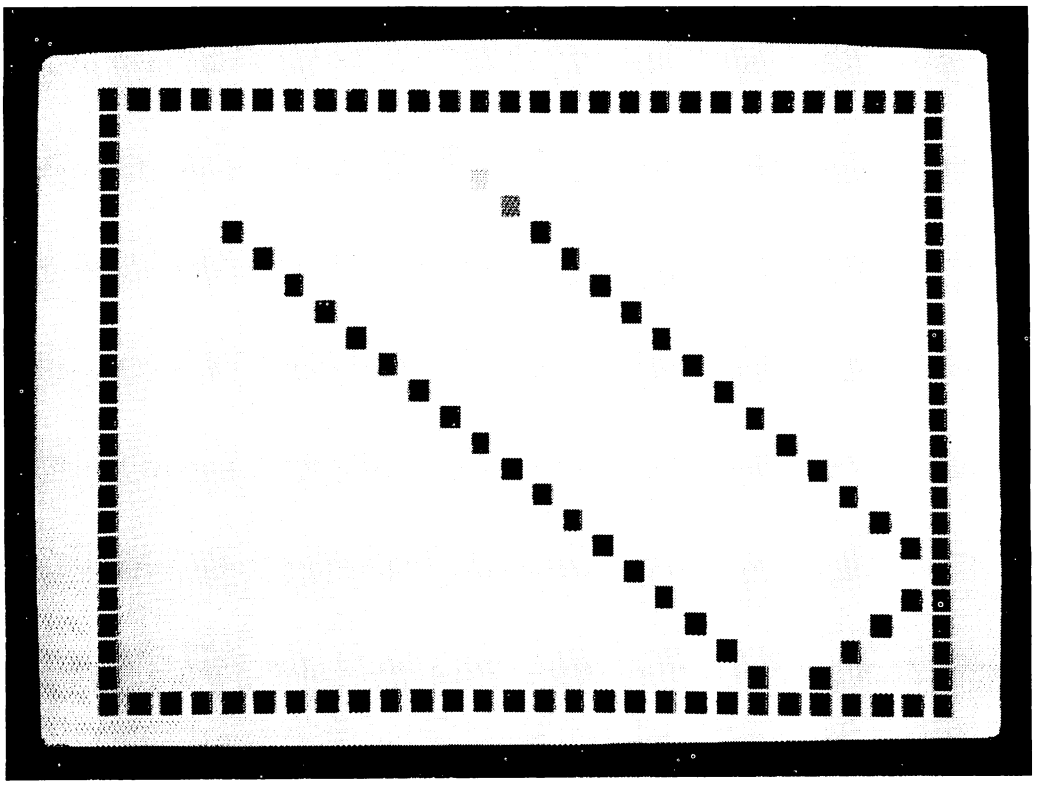
```
10 ! filename:"gr8p1"
20 !  purpose: bouncing dot program
30 !   author: jpg, jdr, & tfz 10/83
40 !
50 DX=1 :: DY=1
60 INPUT "starting coordinates ":X,Y :: CALL CLEAR
70 FOR I=1 TO 28 :: !horizontal boundaries
80 DISPLAY AT(1,I):CHR$(30);:: DISPLAY AT(24,I):CHR$(30);
90 NEXT I
100 FOR I=1 TO 24 :: !vertical boundaries
110 DISPLAY AT(I,1):CHR$(30);:: DISPLAY AT(I,28):CHR$(30);
120 NEXT I
130 DISPLAY AT(Y,X):CHR$(30);:: !put a dot at location X,Y
140 !change direction of dot if necessary
150 IF X=1 OR X=28 THEN DX=-DX
160 IF Y=1 OR Y=24 THEN DY=-DY
170 X=X+DX :: Y=Y+DY :: IF X<=0 THEN X=1 :: IF Y<=0 THEN Y=1
180 GOTO 130
999 END
```

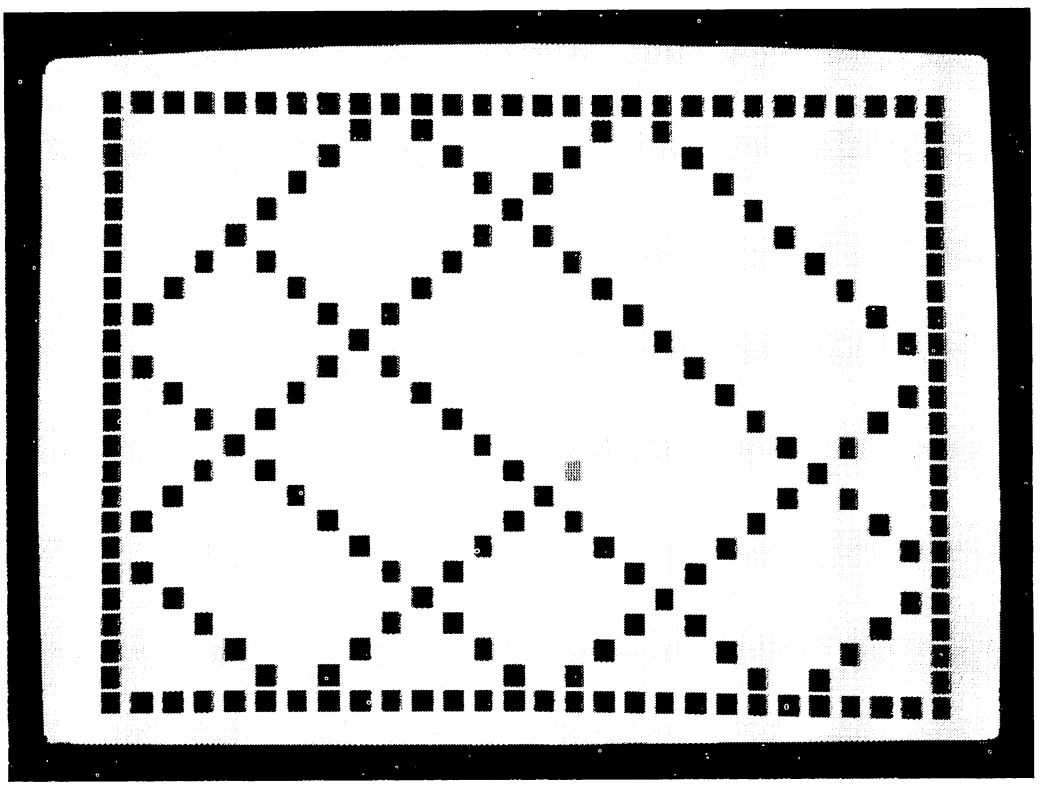
Discussion

- The program defines DX as the value 1 and DY as the value 1 to provide an angled shot at all walls.
- The walls are built on the four sides of the screen.

Suggestions

- Modify the program so that the rectangular area in which the dot bounces can be specified by the user.
- Turn the bouncing dot into a blob and write a program to bounce the blob off horizontal and vertical walls.



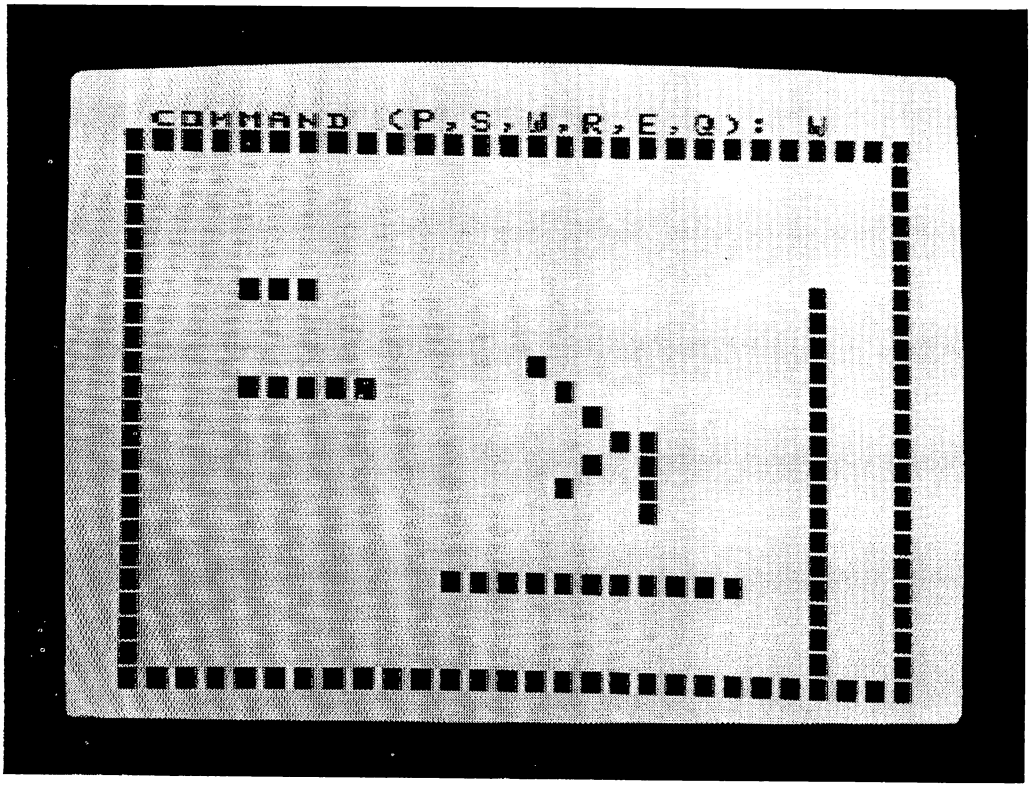


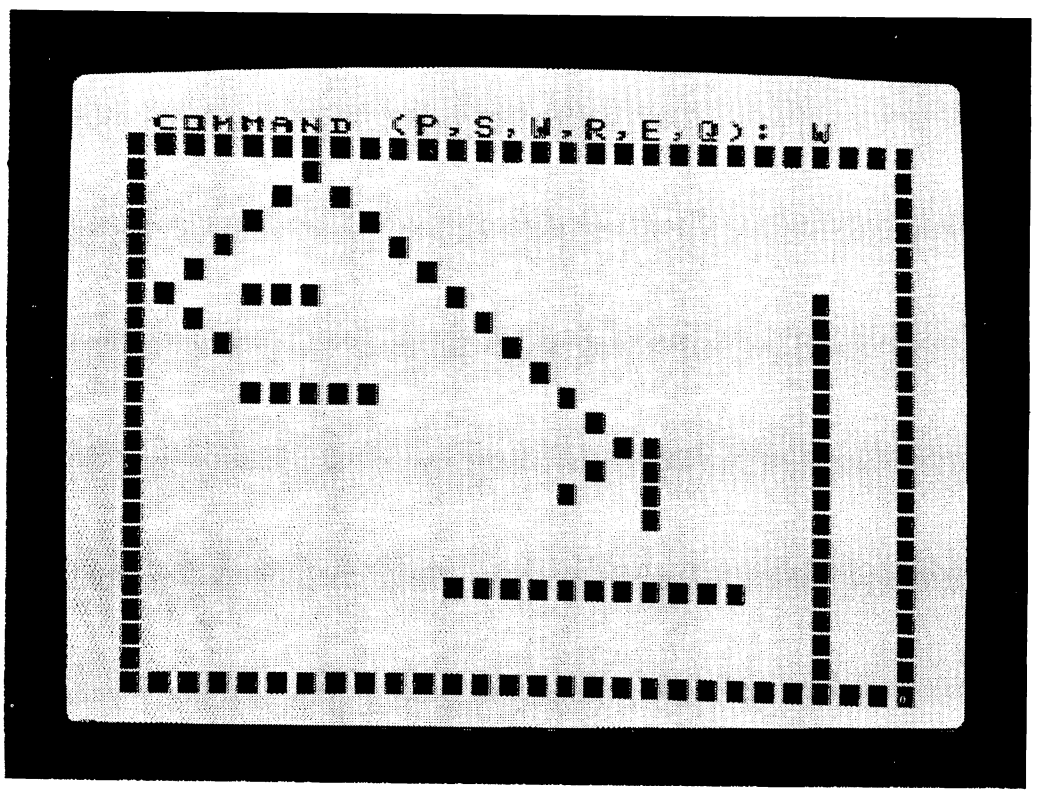
Problem 8.2 Write a program to produce a visual display by bouncing a dot off randomly placed lines on the video screen.

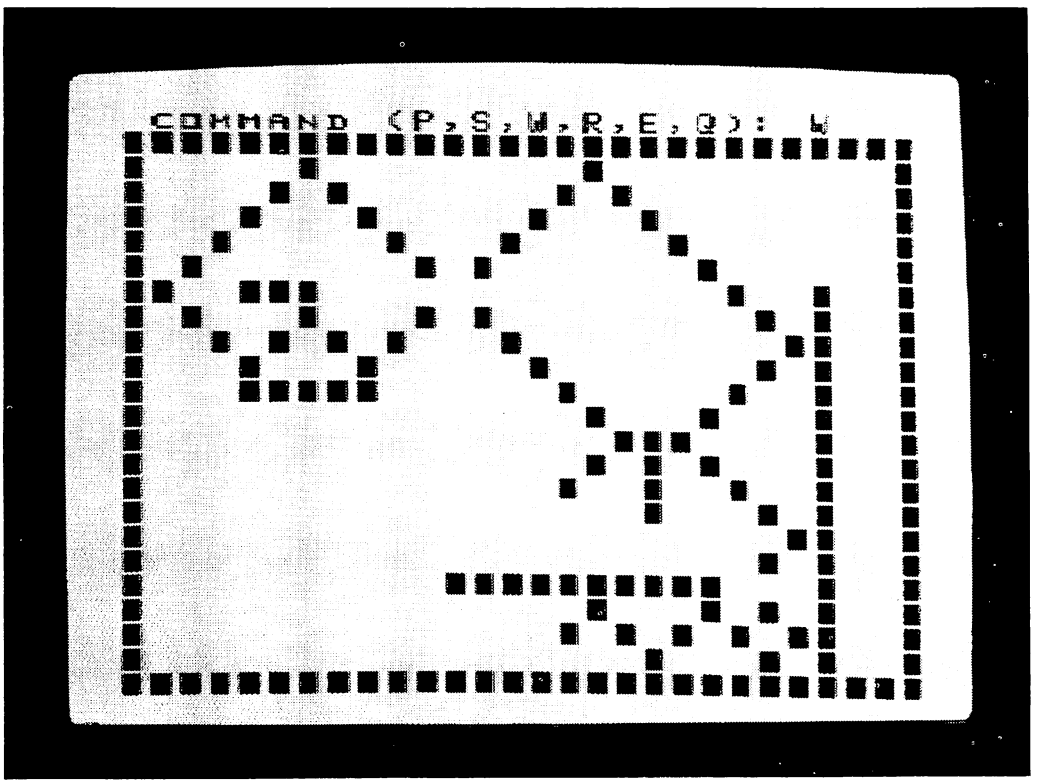
Solution

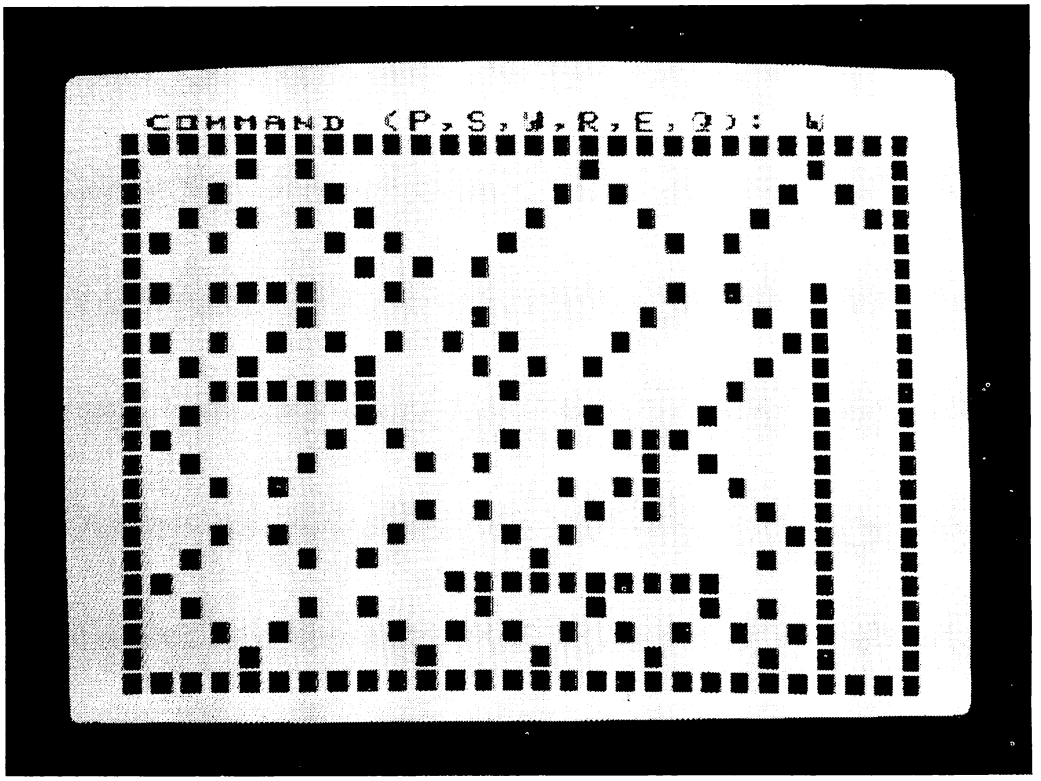
```
10 ! filename:"gr8p2"
20 !  purpose: creating computer art
30 !  author: jpg, jdr, spg, hmz & tfz 10/83
40 !
50 RANDOMIZE
60 !next line initializes variables, clear screen
70 E=1 :: P=-1
80 CALL CLEAR
90 PRINT "  press:" :: PRINT
100 PRINT " L - to look at old pictures"
110 PRINT " M - to create new pictures"
120 PRINT :: INPUT A$
130 IF A$="L" THEN 890
140 !user wants to create new art
150 CALL CLEAR
160 !draw 6 random lines on the screen (3 vert, 3 horiz)
170 C=30
180 FOR L=1 TO 3
190 H0=INT(12*RND)+1 :: V0=2*(1+INT(5*RND))+4
200 H1=H0+INT(12*RND)+1 :: V1=V0+2*(1+INT(11-V0/2)*RND)
210 H2=2*(1+INT(8*RND))+4 :: V2=INT(26*RND)+1
220 FOR N=H0 TO H1 :: XP=N :: YP=H2 :: GOSUB 2000 :: NEXT N
230 FOR N=V0 TO V1 :: XP=V2 :: YP=N :: GOSUB 2000 :: NEXT N
240 NEXT L
250 !initialize the direction of the dot
260 S=INT(RND*2):: IF S=0 THEN S=-1
270 T=INT(RND*2):: IF T=0 THEN T=-1
280 !draw a boundary around the screen
290 FOR N=1 TO 28
300 XP=N :: YP=2 :: GOSUB 2000
310 XP=N :: YP=24 :: GOSUB 2000
320 NEXT N
330 FOR N=2 TO 24
340 YP=N :: XP=1 :: GOSUB 2000
350 YP=N :: XP=28 :: GOSUB 2000
360 NEXT N
370 !start the point where screen is unoccupied
380 C=30
390 YP=2*(1+INT(9*RND))+3 :: XP=2*(1+INT(11*RND))+3
400 CALL GCHAR(YP,XP+2,A):: IF A=30 THEN 390
410 X=XP :: Y=YP
420 !print the command statement on the screen
430 DISPLAY AT(1,2):"command (P,S,W,R,E,Q):";
440 !
450 !change directions if necessary
460 CALL GCHAR(YP,XP+3,A):: IF A=C THEN S=-1
470 CALL GCHAR(YP,XP+1,A):: IF A=C THEN S=1
480 CALL GCHAR(YP+1,XP+2,A):: IF A=C THEN T=-1
490 CALL GCHAR(YP-1,XP+2,A):: IF A=C THEN T=1
500 !blank point X,Y if P=1
510 XP=X :: YP=Y
520 CALL GCHAR(YP,XP+2,A):: IF P=1 AND A=C THEN GOSUB 3000
530 !tell dot where to move
540 IF S=1 THEN X=X+1
550 IF S=-1 THEN X=X-1
560 IF T=1 THEN Y=Y+1
570 IF T=-1 THEN Y=Y-1
580 !erase point if it was filled and E=1
590 XP=X :: YP=Y
```

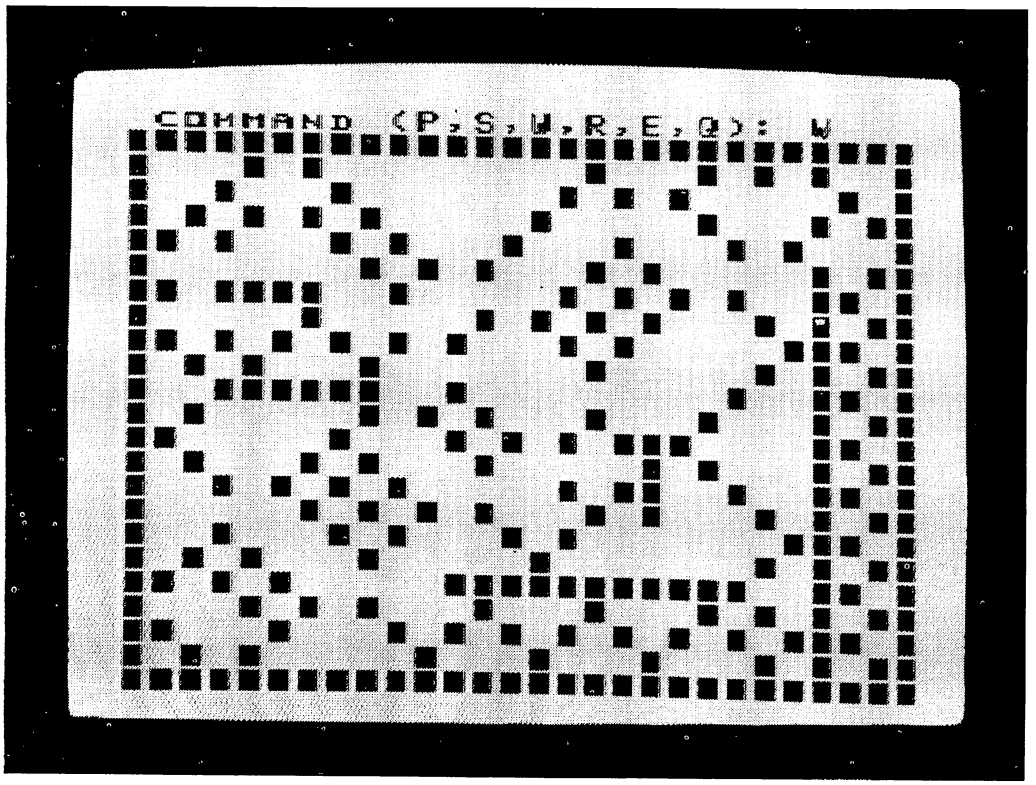
```
600 CALL GCHAR(YP,XP+2,A):: IF A=C AND E=1 THEN GOSUB 3000 ELSE GOSUB 2000
610 !was a key pressed?
620 CALL KEY(0,N,S1):: IF S1=0 THEN 460 :: A$=CHR$(N)
630 !quit?
640 IF A$="Q" THEN CALL CLEAR :: GOTO 9999
650 !print the entered letter within the command
660 DISPLAY AT(1,25):A$
670 !reverse the dot?
680 IF A$="R" THEN T=-T :: S=-S :: GOTO 430
690 !erase the dot's trail?
700 IF A$="P" THEN P=-P :: GOTO 430
710 !erase the dot when it runs over another?
720 IF A$="E" THEN E=-E :: GOTO 430
730 !cause the dot to wait?
740 IF A$<>"W" THEN 770
750 FOR W=1 TO 2000 :: NEXT W :: GOTO 430
760 !store picture on disk?
770 IF A$<>"S" THEN 430
780 OPEN #1:"DSK1.ART",SEQUENTIAL,INTERNAL,APPEND
790 !the next 7 lines copy the screen to disk
800 FOR J=2 TO 24
820 FOR I=3 TO 30
830 CALL GCHAR(J,I,A)
840 PRINT #1:A
850 NEXT I
860 NEXT J
870 CLOSE #1
880 GOTO 430
890 !copies art on disk to screen
900 OPEN #1:"DSK1.ART",SEQUENTIAL,INTERNAL,INPUT
910 IF EOF(1)THEN 1010
920 CALL CLEAR
930 FOR J=2 TO 24
940 DISPLAY AT(1,1):J
950 FOR I=1 TO 28
960 INPUT #1:A
970 DISPLAY AT(J,I):CHR$(A)
980 NEXT I
990 NEXT J
1000 CALL KEY(0,N,S):: IF S=0 THEN 1000 ELSE 910
1010 PRINT "no more art on file..."
1020 FOR I=1 TO 500 :: NEXT I
1030 CALL CLEAR
1040 CLOSE #1
1050 GOTO 90
2000 !set (XP,YP) subroutine
2010 DISPLAY AT(YP,XP):CHR$(30);
2020 RETURN
3000 !reset (XP,YP) subroutine
3010 DISPLAY AT(YP,XP):" ";
3020 RETURN
9999 END
```

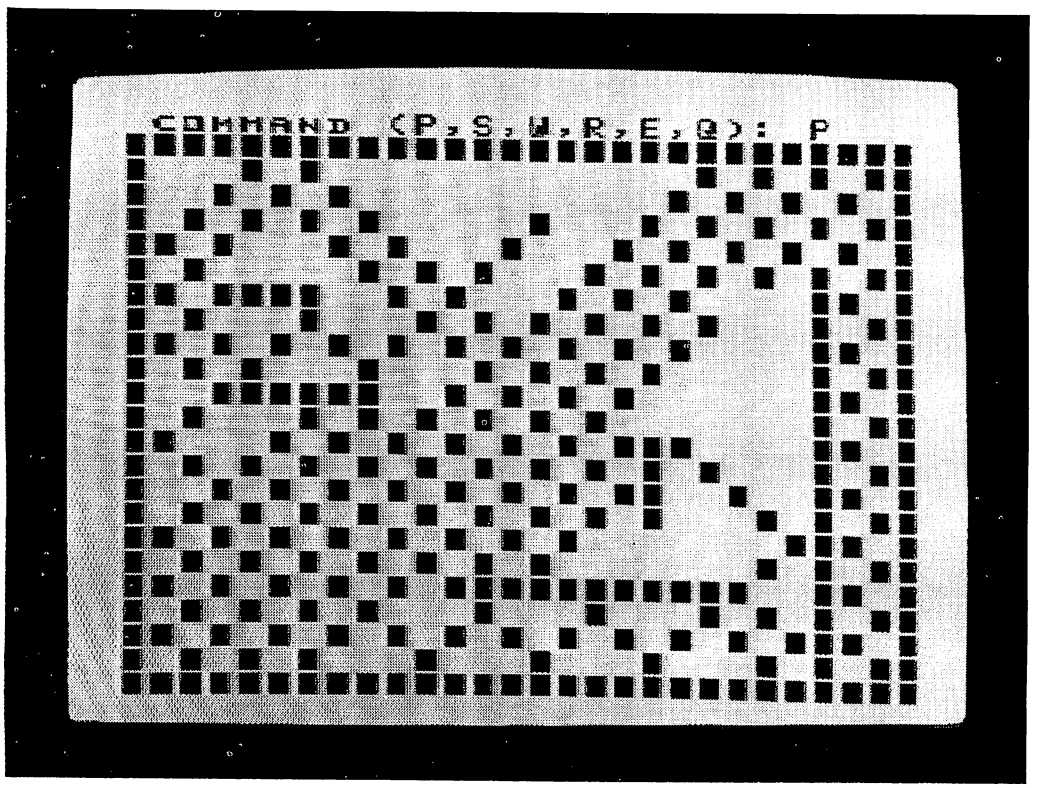

COMMAND (P,S,W,R,E,Q): W


COMMAND (P,S,W,R,E,Q): W


COMMAND (P,S,W,R,E,Q): W


COMMAND (P,S,W,R,E,Q): W


COMMAND (P,S,W,R,E,Q): W


COMMAND (P,S,W,R,E,Q): P

Discussion

- The program allows the user to participate in the way the graphic design proceeds.
  Typing an "R" reverses the dot's direction.
  Typing a "P" causes the dot's trail to be erased.
  Typing an "E" erases a dot that is run over by the moving dot.
  Typing a "W" pauses the dot's motion.
  Typing a "Q" terminates the program.

- The program allows the user to save on disk the screen contents when a particularly interesting pattern is perceived.
  Typing an "S" stores the picture on file.

- All user input is through CALL KEY. Depressing the ENTER key accidentally after typing any of the above keys can cause unpredictable and weird things to happen to the screen pictures.

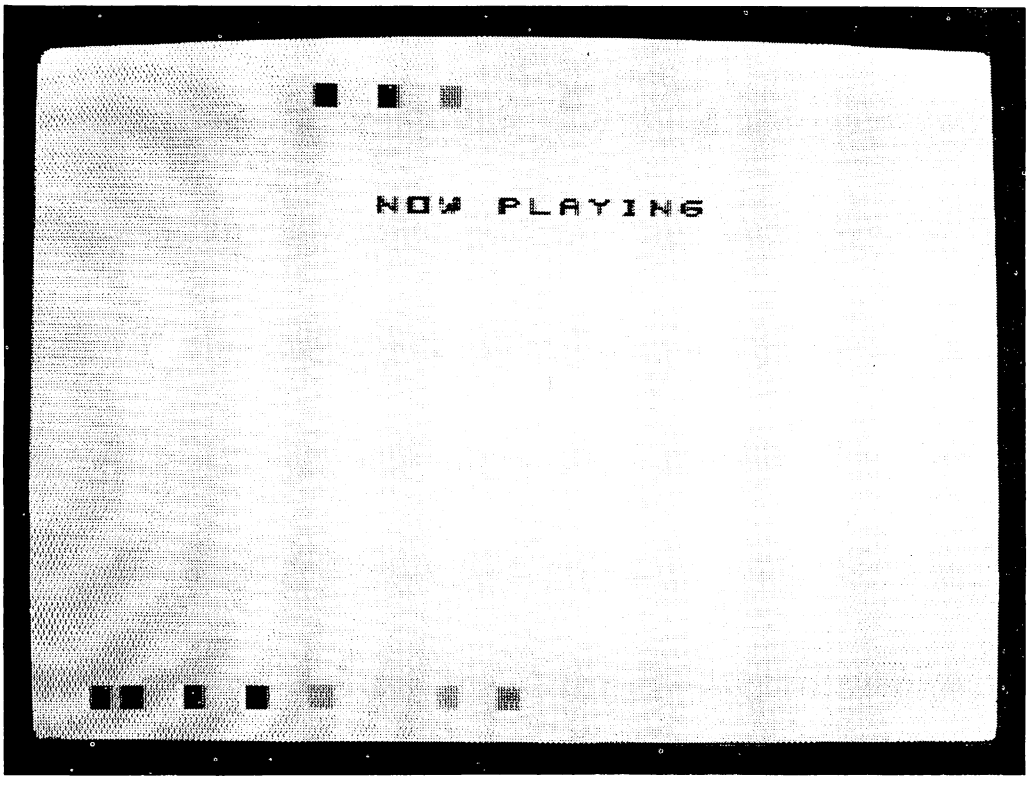
Suggestions

- When a screen picture is saved, the status of the picture is lost and the picture can not be continued. Modify the program so that the dot picks up where it left off when interrupted by the typing of an "S".

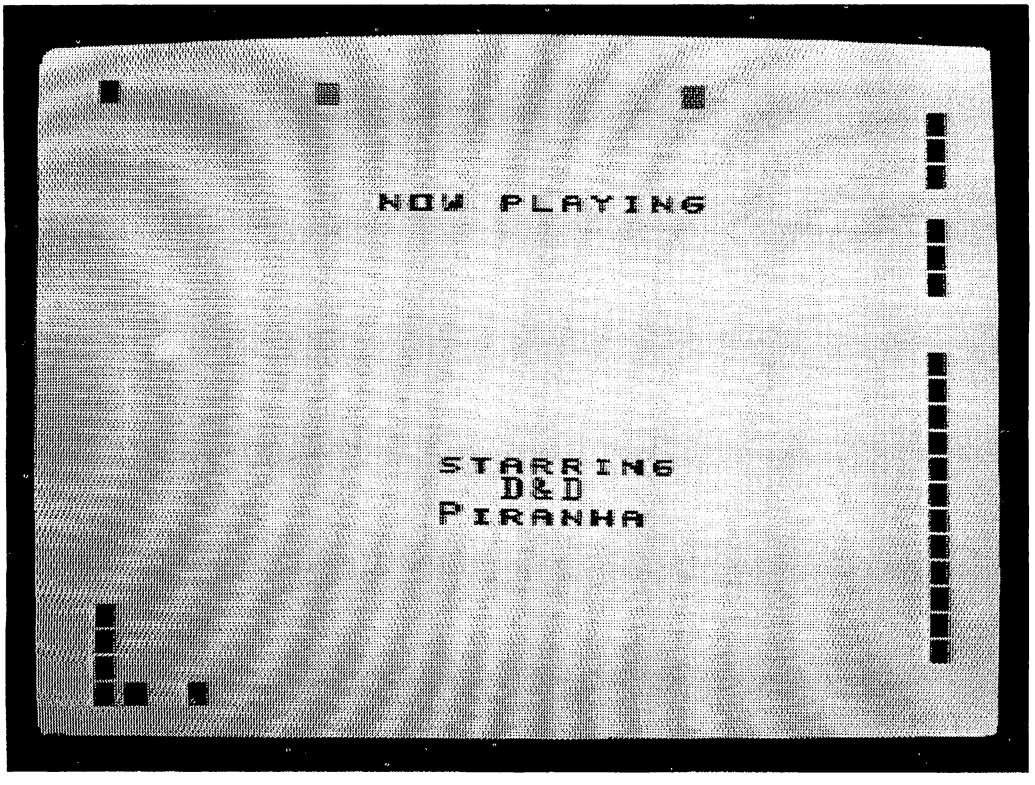
Problem 8.3 Write a program that causes the screen of the TI–99/4A to become a movie marquee hawking the latest flick.

Solution

```
10 ! filename:"gr8p3"
20 !  purpose: marquee display
30 !   author: jpg, jdr & tfz 10/83
40 !
50 CALL CLEAR
70 P$=CHR$(30)
80 READ E$,F$,G$,A$,B$,C$,R$
90 N=20 :: M=5
100 !n=20,m=3 or n=20,m=5 or mod lines 160 & 290 step 2*p
110 !n=20,m=6, modify step in line 160 to be 2*P-1
120 FOR Q=1 TO 100
130 P=Q-INT(Q/M)*M+1
140 IF P=2 THEN DISPLAY AT(9,7):R$&R$&R$;ELSE DISPLAY AT(9,7):F$;
150 IF P=2 THEN DISPLAY AT(10,9):R$&R$&R$;ELSE DISPLAY AT(10,9):G$;
160 FOR I=0 TO 28 STEP P
170 J1=24-(I-INT(I/24)*24):: J3=28-J1 :: J2=24-J1 :: J4=24+J1
180 DISPLAY AT(1,J1):P$ :: DISPLAY AT(1,J3):P$
190 DISPLAY AT(24,J2):P$ :: DISPLAY AT(24,J4):P$
200 IF (I-INT(I/5)*5)<>0 THEN DISPLAY AT(5,10):E$;ELSE DISPLAY AT(5,10):R$&R$;
210 IF J1<18 THEN DISPLAY AT(1,J1+4):" ";ELSE DISPLAY AT(1,J1-16):" ";
220 IF J2>4 THEN DISPLAY AT(12,J2-4):" ";ELSE DISPLAY AT(12,J2+16):" ";
230 IF J3>21 THEN DISPLAY AT(1,J3-4):" ";ELSE DISPLAY AT(1,J3+16):" ";
240 IF J4<23 THEN DISPLAY AT(23,J4-16):" ";ELSE DISPLAY AT(23,J4-16):" ";
250 NEXT I
260 IF P=1 THEN DISPLAY AT(15,12):A$;ELSE DISPLAY AT(15,12):R$;
270 IF P=1 THEN DISPLAY AT(16,14):B$;ELSE DISPLAY AT(14,19):R$;
280 IF P=1 THEN DISPLAY AT(17,11):C$;ELSE DISPLAY AT(17,11):R$;
290 FOR J=P TO 24 STEP P
300 DISPLAY AT(J,1):P$;:: DISPLAY AT(J,28):P$
310 FOR K=1 TO N :: NEXT K
320 IF J>5 THEN DISPLAY AT(J-4,1):" ";:: DISPLAY AT(J-4,27):" ";
330 IF J<=4 THEN DISPLAY AT(J+10,1):" ";:: DISPLAY AT(J+10,27):" ";
340 NEXT J
350 NEXT Q
360 DATA "now playing"
370 DATA "The Hedgehog that"
380 DATA "Ate Cambridge"
390 DATA "starring","D&D"," Piranha"
400 DATA "         "
410 END
```


NOW PLAYING


NOW PLAYING
THE HEDGEHOG THAT
ATE CAMBRIDGE


NOW PLAYING
STARRING
D&D
PIRANHA


NOW PLAYING
THE HEDGEHOG THAT
ATE CAMBRIDGE
STARRING
D&D
PIRANHA

Discussion

- The marquee lights cycle from the center top toward the edges, then down the sides where they turn to converge at the center bottom of the screen.
- Some parameters have been built in to control speed, pattern, and cycle of the marquee lights.
- The movie's title and cast come and go giving added movement to the marquee.

Suggestions

- Run the program varying the parameters to get a feel for the type of visual displays that can be produced by the marquee.
- Experiment with the controlling loop structures to change the display. Some simple changes produce remarkable and unpredictable results.
- Change the size of the marquee.

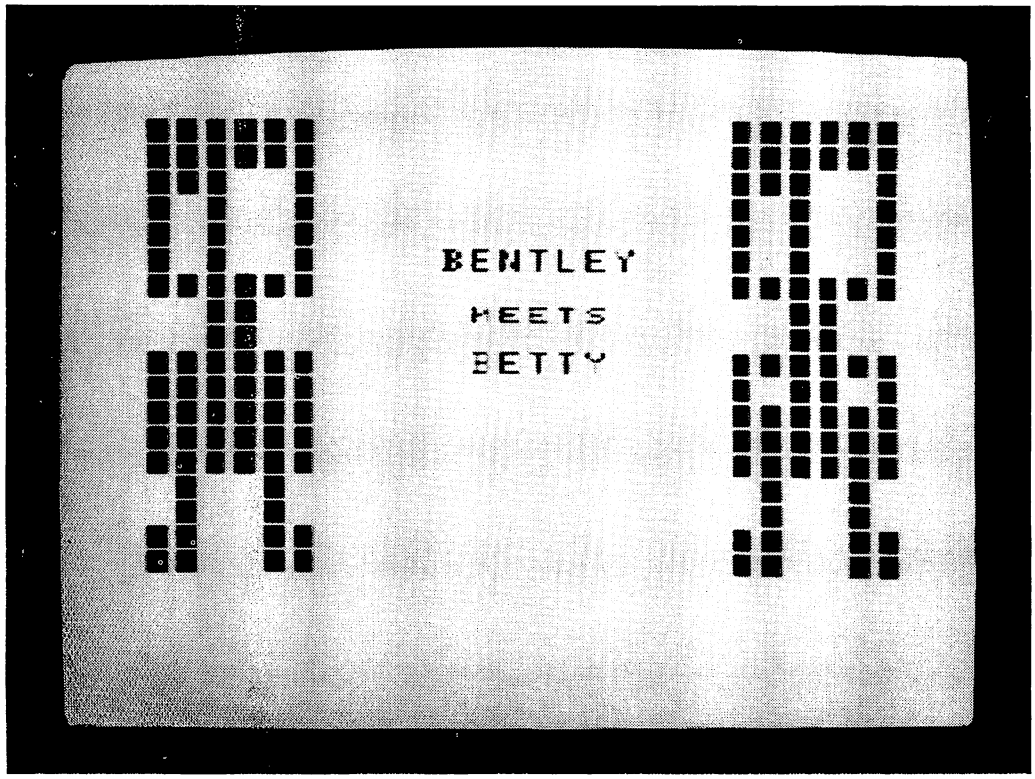
Problem 8.4

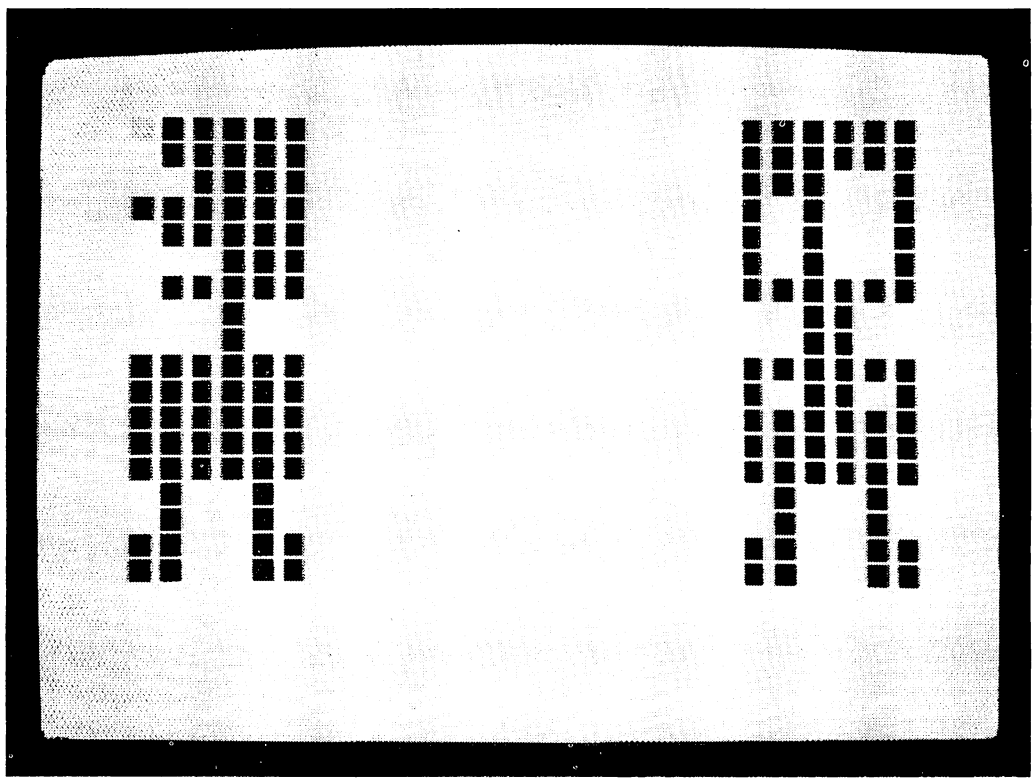
Write a program that will cause stick figures to meet and fall in love.

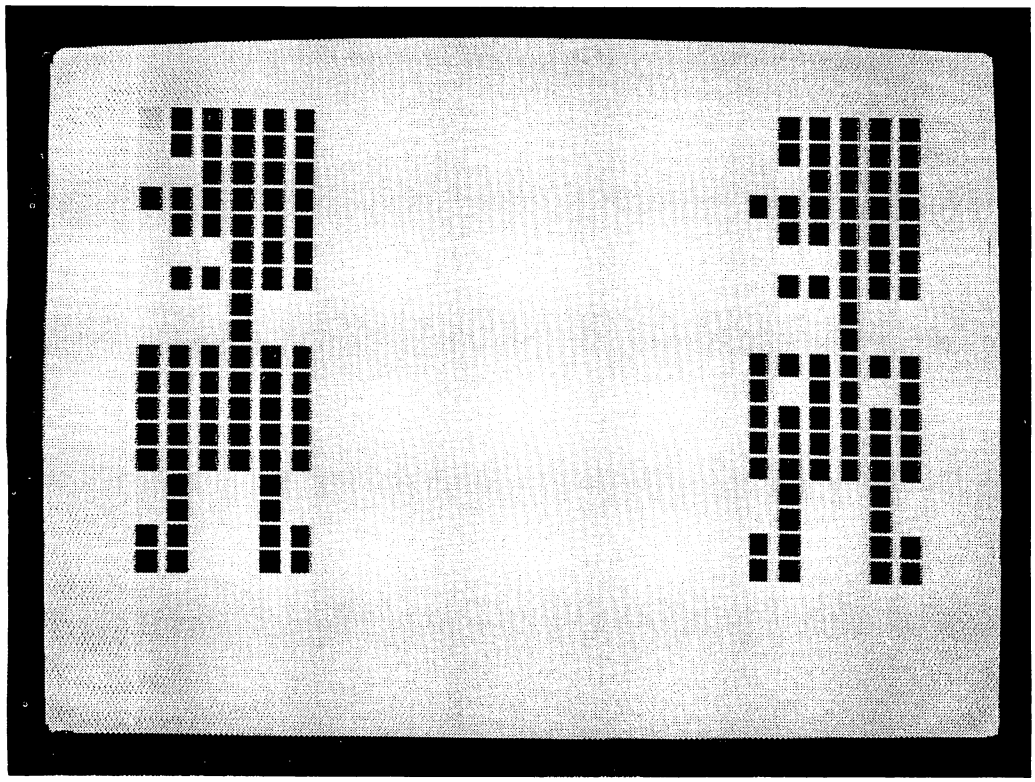
Solution

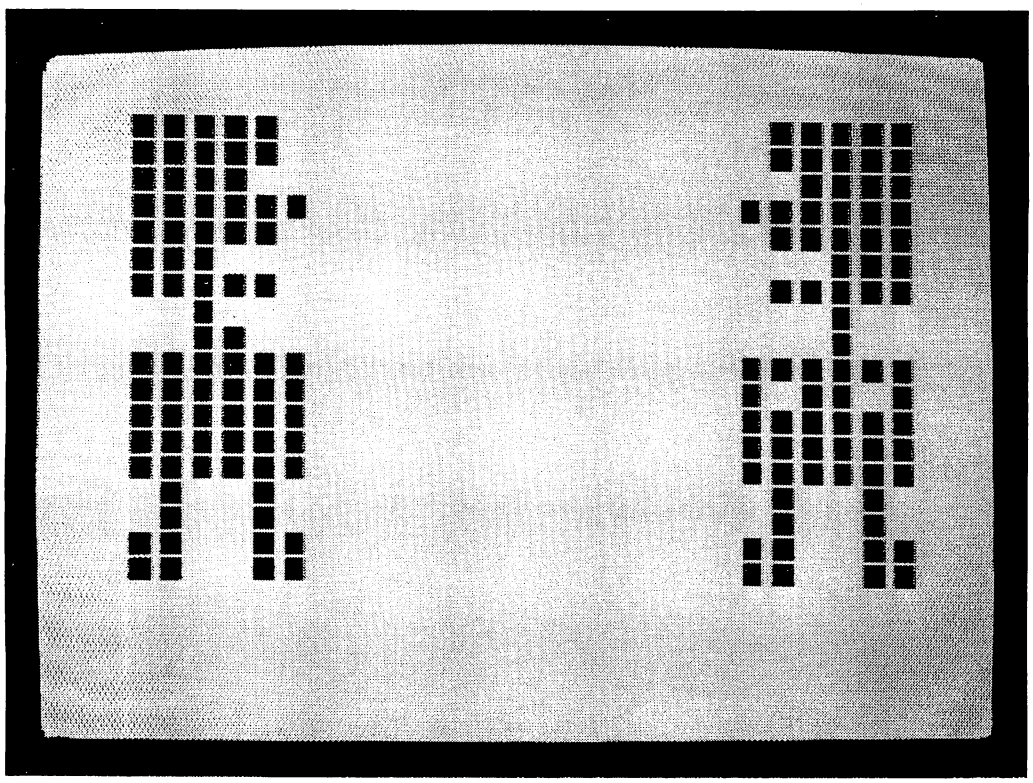
```
10 ! filename:"gr8p4"
20 !  purpose: Bentley meets Betty
30 !   author: jpg, jdr & tfz 10/83
40 !
50 RANDOMIZE
60 DIM F(108),P(108),C$(2)
70 C$(1)=CHR$(30):: C$(0)=" "
80 FOR I=1 TO 108 :: !input head ad body facing front
90 READ F(I)
100 NEXT I
110 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,0,1
120 DATA 1,0,1,0,0,1,1,0,1,0,0,1,1,0,1,0,0,1
130 DATA 1,1,1,1,1,1,1,0,0,1,1,0,0,0,0,1,1,0,0
140 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
150 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,0,0,1,0
160 DATA 0,1,0,0,1,0,1,1,0,0,1,1,1,1,0,0,1,1
170 FOR I=1 TO 108 :: !input head and body left(or right) profile
180 READ P(I)
190 NEXT I
200 DATA 0,1,1,1,1,1,1,0,1,1,1,1,1,1,0,0,1,1,1,1
210 DATA 1,1,1,1,1,1,1,0,1,1,1,1,1,1,0,0,0,1,1,1
220 DATA 0,1,1,1,1,1,1,0,0,0,1,0,0,0,0,0,1,0,0
230 DATA 0,1,1,1,1,1,1,0,1,1,1,1,1,1,0,1,1,1,1,1
240 DATA 0,1,1,1,1,1,1,0,1,1,1,1,1,1,0,0,0,1,1,0
250 DATA 0,0,0,1,1,0,0,1,0,1,1,0,0,1,1,1,1,0
260 CALL CLEAR
270 X=2 :: Y=2 :: H=0 :: GOSUB 1000 :: !draw Bentley
280 X=22 :: Y=2 :: H=0 :: GOSUB 2000 :: !draw Betty
290 DISPLAY AT(7,12):"BENTLEY";
300 DISPLAY AT(9,13):"meets";
310 DISPLAY AT(11,13):"BETTY";
320 D1=-1 :: D2=-1
330 FOR I=1 TO 1000 :: NEXT I
```

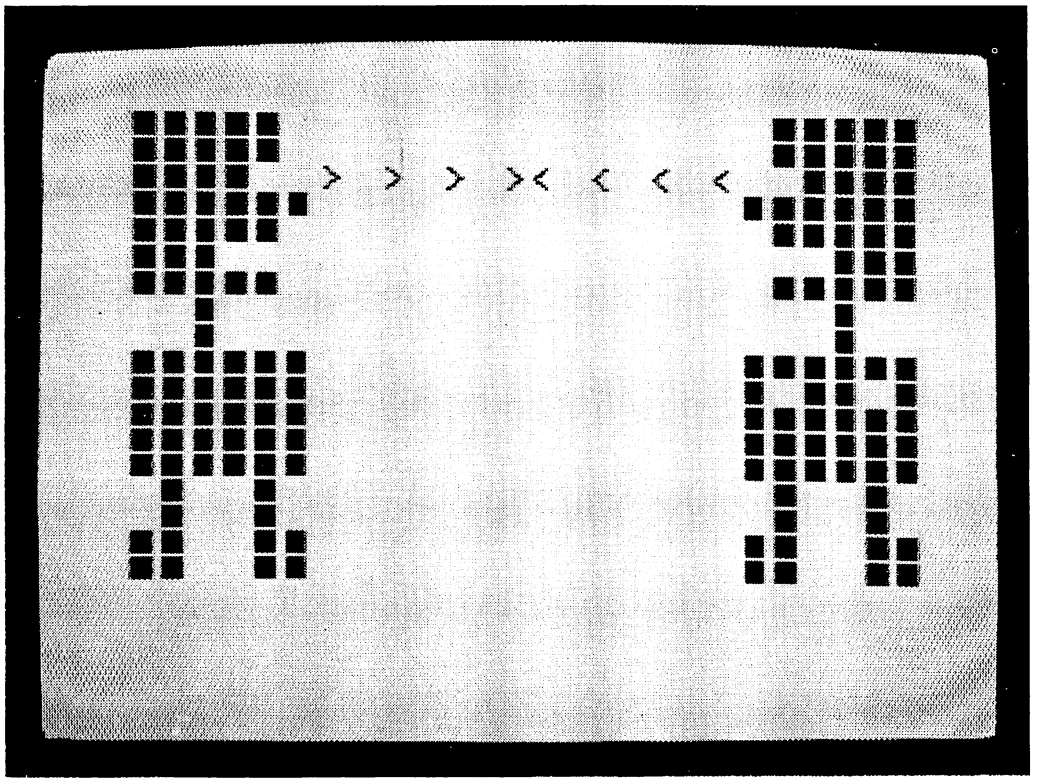
```
340 FOR I=7 TO 11 STEP 2
350 DISPLAY AT(I,12):"        ";
360 NEXT I
370 Y=2 :: D=INT(2*RND):: H=1
380 IF RND>.5 THEN X=2 ELSE X=22
390 IF X=2 THEN IF D1<0 THEN D1=D ELSE D1=-1
400 IF X=2 AND D1=D THEN GOSUB 3000
410 IF X=2 AND D1=-1 THEN GOSUB 1000
420 IF X=22 THEN IF D2<0 THEN D2=D ELSE D2=-1
430 IF X=22 AND D2=D THEN GOSUB 3000
440 IF X=22 AND D2=-1 THEN GOSUB 1000
450 FOR I=1 TO 50 :: NEXT I
460 IF D1=1 AND D2=0 THEN GOSUB 5000 ELSE 370
470 CALL CLEAR
480 STOP
1000 !draw Bentley face front at X,Y
1010 K=0
1020 IF H=0 THEN I4=17 ELSE I4=8
1030 FOR J=Y TO Y+I4
1040 FOR I=0 TO 5
1050 K=K+1
1060 DISPLAY AT(J,X+I):C$(F(K));
1070 NEXT I
1080 NEXT J
1090 RETURN
2000 !draw Betty face front at X,Y
2010 GOSUB 1000 :: !draw Bentley face front
2020 DISPLAY AT(Y+10,X+1):C$(0);
2030 DISPLAY AT(Y+10,X+4):C$(0);
2040 RETURN
3000 !ddraw Bentley left(or right) profile at X,Y
3010 K=0
3020 IF D=0 THEN I1=0 :: I2=5 :: I3=1 ELSE I1=5 :: I2=0 :: I3=-1
3030 !D=0 ->left, D=1 ->right
3040 IF H=0 THEN I4=17 ELSE I4=8
3050 !H=0 ->head & body, H=1 ->head only
3060 FOR J=Y TO Y+I4
3070 FOR I=I1 TO I2 STEP I3
3080 K=K+1
3090 DISPLAY AT(J,X+I):C$(P(K));
3100 NEXT I
3110 NEXT J
3120 RETURN
4000 !draw Betty left(or right) profile at X,Y
4010 GOSUB 3000 :: !draw Bentley profile
4020 IF D=0 THEN I4=0 ELSE I4=5
4030 DISPLAY AT(Y+10,X+I4):C$(1);
4040 RETURN
5000 !love at first sight
5010 FOR I=8 TO 15 STEP 2
5020 DISPLAY AT(Y+2,I):">";
5030 DISPLAY AT(Y+2,29-I):"<";
5040 NEXT I
5050 X=2 :: Y=2 :: H=0 :: D=D1 :: GOSUB 3000
5060 X=22 :: Y=2 :: H=0 :: D=D2 :: GOSUB 4000
5070 FOR I=1 TO 1000 :: NEXT I
5080 DISPLAY AT(Y+2,8):RPT$(CHR$(32),15);
5090 FOR I=8 TO 15 STEP 2
5100 DISPLAY AT(Y+10,I):CHR$(42);
5110 DISPLAY AT(Y+10,28-I):CHR$(42);
5120 NEXT I
5130 FOR I=1 TO 2000 :: NEXT I
5140 RETURN
9999 END
```

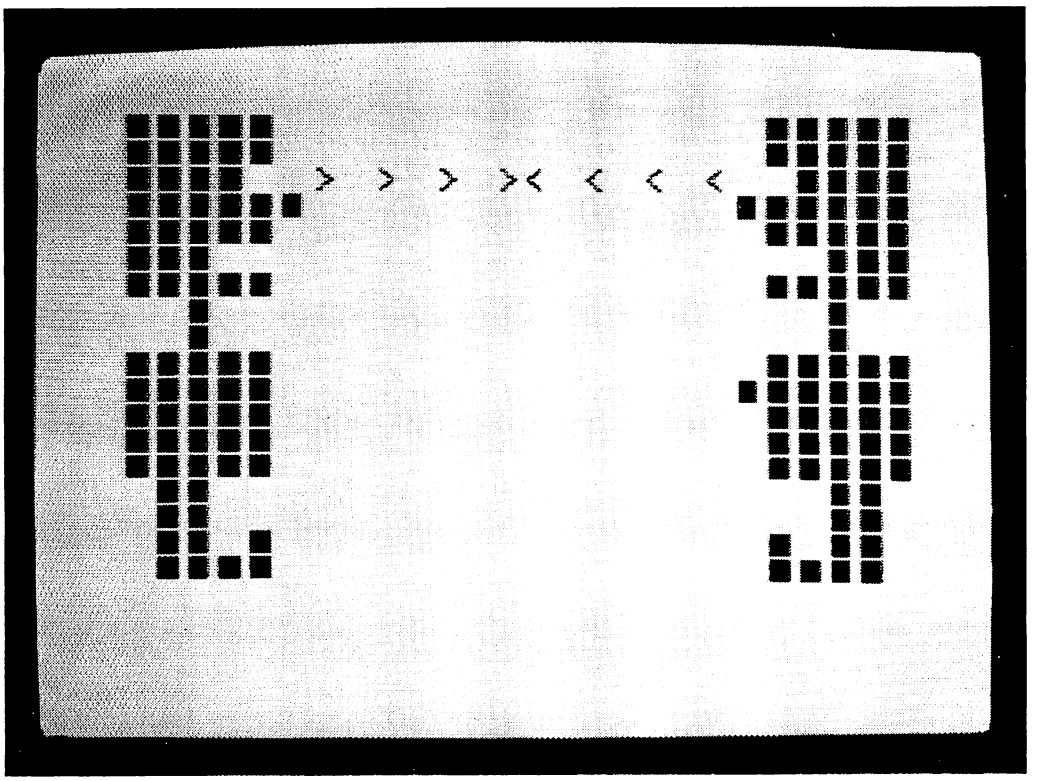

BENTLEY
MEETS
BETTY

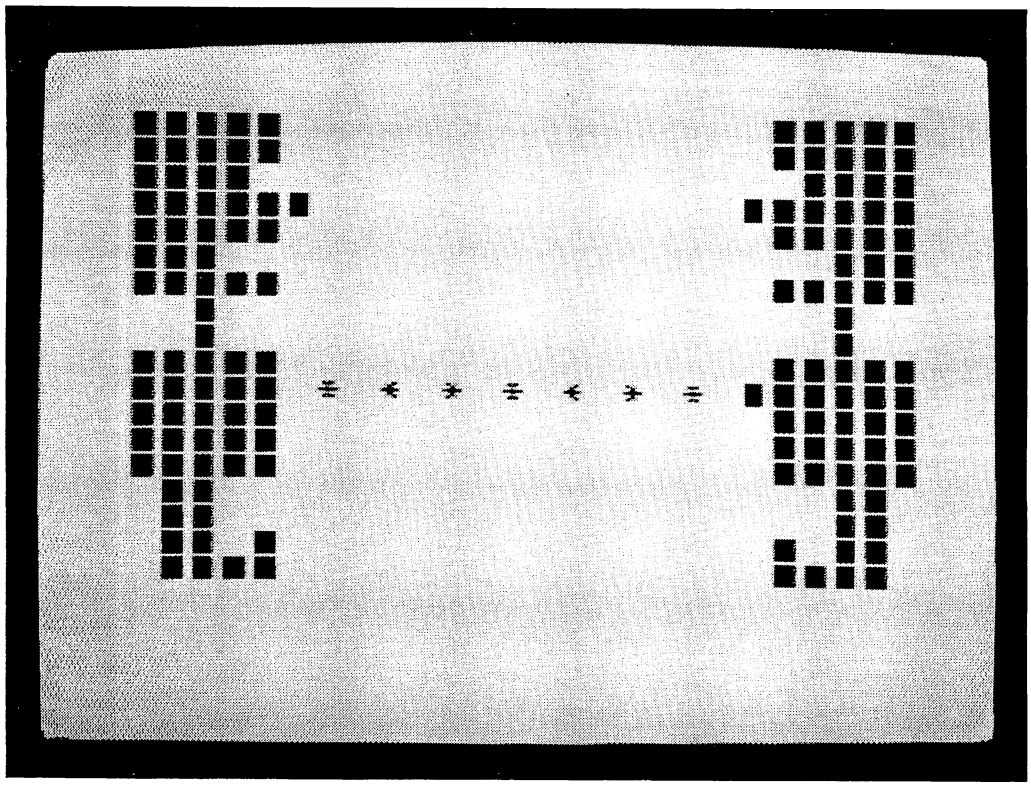












Discussion

- This program is an example of the kind of simple animation that can be programmed easily.
- The subroutines 1000 and 2000 draw Bentley and Betty's bodies with their heads facing forward.
- The profiles are changed by the routines at 3000 and 4000, which are randomly selected to make them face left or right until both face each other.
- The tables F and P contain the Front and Profile information processed similar to the large digits in Chapter 6.

Suggestions

- When Bentley meets Betty, have them walk toward one another.
- Make them blink or open and close their mouths.
- Make both figures levitate with love.

TO OUR READERS. . . .

Would you like to know more about microcomputer books published by Wm. C. Brown? If so, please provide the following information:

Name _____

Company (if applicable) _____

Street _____

City _____ State _____ Zip _____

Most titles are available through better book and computer stores.

READER'S CRITIQUE. . . .

Wm. C. Brown wants to publish microcomputer books that meet your needs. Your feedback therefore is vital for us to improve current books and publish new and even better books in the future.

The feedback we need from you is outlined in the following questions. This page, when folded along the dotted lines in thirds, can be mailed back to us at no cost to you. Thank you in advance for the valuable information you are providing.

1. In which Wm.C. Brown microcomputer book did you find this form?
   Author: _____ Title: _____
2. Please rate this book in the following areas:

	Excellent	Good	Fair	Poor
a. Clarity of information	___	___	___	___
b. Completeness of explanations	___	___	___	___
c. Quantity of Examples	___	___	___	___
d. Usefulness of Examples	___	___	___	___
e. Degree to which Book Met Your Expectations	___	___	___	___
f. Overall Rating of Book	___	___	___	___

3. From what source did you purchase this book?
   ( ) Bookstore ( ) Convention Order ( ) Directly from Wm. C. Brown
   ( ) Computer Store ( ) Other: _____
4. What brand(s) and model(s) of microcomputer(s) do you own/use?
   _____
5. What computer language(s) do you use? _____
6. For how many years have you been using a computer?
   ( ) Less than 1 year ( ) 1-2 years ( ) 3-5 years ( ) More than 5 years
7. Which, if any, computer magazines/newspapers do you read regularly?
   _____
8. On what topics would you be most interested in purchasing microcomputer books? _____
   _____
9. Please suggest ways in which WCB can improve future microcomputer books we publish.
   _____

QUESTIONS #10-14 ARE MERELY FOR CLASSIFICATION AND WILL BE USED ONLY TO DEVELOP A STATISTICAL PROFILE OF OUR RESPONDENTS.

10. Your occupation/job title? _____
11. Do you use a computer in your work? ( ) Yes ( ) No
12. Your age?
    ( ) Under 18 ( ) 25-29 ( ) 40-49 ( ) 65 and over
    ( ) 18-24 ( ) 30-39 ( ) 50-64
13. Are you male or female? ( ) Male ( ) Female
14. What is the highest level of education you have attained?
    ( ) 8th grade or less ( ) High school graduate ( ) College graduate
    ( ) Attended/attending high school ( ) Attended/attending college ( ) Post graduate study

"THANK YOU"

wcb

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 551 DUBUQUE, IOWA 52001

POSTAGE WILL BE PAID BY ADDRESSEE

WM. C. BROWN PUBLISHERS
Marketing Research Department
2460 Kerper Boulevard
P.O. Box 539
Dubuque, Iowa 52001

ABOUT THE AUTHORS

John P. Grillo

Dr. Grillo earned his Ph.D. in 1971 from the University of New Mexico. His professional experience includes consulting in the areas of word processing, mail-order systems, structured programming techniques, and database systems. He is a contributor to trade and professional journals, as well as coauthor of numerous books. He is a member of the Bentley College faculty in the Computer Information Systems Department.

J. D. Robertson

Dr. Robertson earned his Ph.D. in Computer Science in 1976 from the University of Southwestern Louisiana. He has been a consultant for book publishers, state agencies, various small businesses, and computerized farm management systems. He has written articles for business and professional journals and coauthored numerous books. He is a member of the Bentley College faculty in the Computer Information Systems Department.

Terry F. Zbyszynski

Terry Zbyszynski received her B.A. from Harvard and her Masters in Education from Lesley College in Cambridge, Massachusetts. She is currently the Computer Coordinator for Lanmark School in Prides Crossing, Massachusetts.

wcb
Wm. C. Brown Publishers
Dubuque, Iowa

$14.95

ISBN 0-697-00237-3